





This is to certify that the

thesis entitled

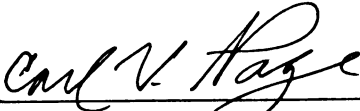
DEVELOPING A COMPUTER PROGRAM IN AN ARTIFICIAL  
INTELLIGENCE LANGUAGE FOR PAVEMENT DESIGN  
INCLUDING MATERIAL AND COST ESTIMATION

presented by

Muhammad Bashir

has been accepted towards fulfillment  
of the requirements for

M.S. degree in Computer Science

  
Major professor

Date 12/18/95



**LIBRARY**  
**Michigan State**  
**University**

**PLACE IN RETURN BOX** to remove this checkout from your record.  
**TO AVOID FINES** return on or before date due.

DATE DUE	DATE DUE	DATE DUE
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

MSU Is An Affirmative Action/Equal Opportunity Institution

c:\pic\data\due.pm3-p.1

## ABSTRACT

### DEVELOPING A COMPUTER PROGRAM IN ARTIFICIAL-INTELLIGENCE LANGUAGE FOR PAVEMENT DESIGN INCLUDING MATERIAL AND COST ESTIMATION

### DEVELOPING A COMPUTER PROGRAM IN AN ARTIFICIAL INTELLIGENCE LANGUAGE FOR PAVEMENT DESIGN INCLUDING MATERIAL AND COST ESTIMATION

A computer program named BASHROAD has been developed in LISP Programming  
(LISP) Language for designing different layers of flexible pavement, using the AASHTO

(American Association of State, Highway and Transportation Officials) procedure for  
flexible pavement thickness design. It calculates material quantities required and

the cost of materials and road section. This system consists of a production rule system  
(containing a structure to store the rules, a production rule memory, a production rule  
handler etc.), a set of production rules stored in the production rule memory, and a set of  
functions. The functions are used for obtaining input data from the user and to check its

validity. The production rules are designed to show and display important results on

the screen including values of different material quantities, different layer  
thickness of the road, material cost, and road section cost and  
the cost per lane mile and for entire road section.

The program has been tested by using different input data and giving accurate results.

Overall working of this system is also explained with the help of some diagrams.

This system is designed in order to assist the engineer in the design of flexible pavement system,

using Artificial Intelligence, in the future. It is expected that such a system would  
be to find the best suitable alignment on the right-of-way and design of the road at any  
two places such that maximum job can be done within the budget.

1995

Department of Computer Science

## **ABSTRACT**

### **DEVELOPING A COMPUTER PROGRAM IN ARTIFICIAL INTELLIGENCE LANGUAGE FOR PAVEMENT DESIGN INCLUDING MATERIAL AND COST ESTIMATION**

**By**

**Muhammad Bashir**

A computer program named BASHROAD has been developed in List Programming (LISP) Language for designing different layers of flexible pavement, using the AASHTO (American Association of State, Highway, and Transportation Officials) procedure for flexible pavement thickness design, and for calculation of material quantities required and the cost of materials and road section. This system consists of a production rule system (containing a structure to store the rules, a production rule memory, a production rule handler etc.), a set of production rules stored in the production rule memory, and a set of functions. The functions are used for obtaining input data from the user and to check its validity. The production rules are designed to calculate and display important results on the screen including values of certain design parameters/variables, different layer thickness of the road, material quantities required for its construction, material cost and the cost per lane mile and for entire road section.

The program has been tested by using different data and gives accurate results. Overall working of this system is also elaborated with the help of some examples.

This system is designed in order to become a part of an anticipated expert system, using Artificial Intelligence, in the future. The ultimate aim of this expert system would be to find the best suitable alignment on the digital map and design a road between any two places such that maximum job can be done just by using this expert system.

## **ACKNOWLEDGMENTS**

First of all, thanks to Allah Almighty for providing me the opportunity to undertake and complete this task. I also want to thank the Pakistan Army Corps of Engineers and the National University of Science and Technology (NUST) for sponsoring my study program.

I want to express my gratitude and sincere thanks to my advisor, Dr. Carl V Page for **TO MY PARENTS, WIFE AND FAMILY** have been possible to pursue and complete this study. He spared time and guided me whenever I needed to be guided.

Dr. G. Y. Baladi and Dr. W. C Tayler deserve special thanks for serving on the advisory committee for this study and for their comments and interest. They provided the professional knowledge and guidance in the field of road designing by answering my queries, which made the job of programming easier.

I acknowledge with gratitude the dedicated and professional assistance provided by Dr. William F. Punch and Anwar Ul Haq Chaudhary. Dr. William F. Punch guided me to handle the overall programming and Anwar Ul Haq Chaudhary assisted my queries regarding road designing in the absence of my other professor. Mr. George C. Stockman deserves special thanks for taking keen interest in this study and providing moral support at each stage.



## ACKNOWLEDGMENTS

Dr. Jamil Sawar (NUST, Pakistan) deserve special thanks for providing me knowledge about the List Programming (LISP) Language which enabled me to accomplish this task successfully.

First of all, thanks to Allah Almighty for providing me the opportunity to undertake and complete this task. I also want to thank the Pakistan Army Corps of Engineers and the National University of Science and Technology (NUST) for inspiration. My daughters Aleena and Hanna deserve special thanks for providing sponsoring my study program.

I want to express my gratitude and sincere thanks to my advisor, Dr. Carl V Page for his continuous professional guidance and moral support without which it would not have been possible to pursue and complete this study. He spared time and guided me support which proved to be a source of encouragement and inspiration for me, without whenever I needed to be guided.

Dr. G. Y. Baladi and Dr. W. C Taylor deserve special thanks for serving on the advisory committee for this study and for their comments and interest. They provided the professional knowledge and guidance in the field of road designing by answering my queries, which made the job of programming easier.

I acknowledge with gratitude the dedicated and professional assistance provided by Dr. William F. Punch and Anwar Ul Haq Chaudhary. Dr. William F. Punch guided me to handle the overall programming and Anwar Ul Haq Chaudhary answered my queries regarding road designing in the absence of any other professor. Dr. George C. Stockman deserves special thanks for taking keen interest in this study and providing moral support at each stage.



Dr Jamil Sawar and Military College of Signals, (NUST, Pakistan) deserve special thanks for providing me knowledge about the List Programming (LISP) Language which enabled me to accomplish this task successfully.

I am thankful to my wife for her sacrifices, patience, understanding and continuous prayers at difficult moments. This was a source of encouragement and inspiration. My daughters Aleena and Hamna deserve special thanks for providing moments of pleasure and relief, which were essentially needed.

I am grateful to my mother, sisters, brothers and my in-laws, distinctively, my father in law professor Chaudhary Nawab Ud Din for their sacrifices, prayers, and moral support which proved to be a source of encouragement and inspiration for me, without which it could have been unattainable task to pursue and complete this study.

1.4.2	Designing of Roads .....	6
1.4.3	Material estimation .....	7
1.4.4	Cost estimation .....	8

## CHAPTER 2 PRODUCTION RULE SYSTEM .....

2.1	GENERAL .....	9
2.2	STRUCTURE TO STORE THE RULE .....	9
2.2.1	priority .....	9

## TABLE OF CONTENTS

### **CHAPTER 1**

#### **INTRODUCTION**

1.1	GENERAL	10
1.2	OVERALL VISUALIZATION OF PROBLEM	13
1.3	RECOMMENDED/FOLLOWED APPROACH	34
1.4	PROBLEMS ADDRESSED	54
1.4.1	Designing of a production rule system	54
1.4.2	Designing of Roads	65
1.4.3	Material estimation	76
1.4.4	Cost estimation	8

### **CHAPTER 2**

#### **PRODUCTION RULE SYSTEM**

2.1	GENERAL	9
2.2	STRUCTURE TO STORE THE RULE	9
2.2.1	priority	9

3.3	2.2.2 rule-name .....	9
3.4	2.2.3 arrow .....	9
	2.2.4 s-exp .....	10
	2.2.5 pattern .....	10
2.3	FUNCTION OF PRODUCTION RULE SYSTEM .....	10
	2.3.1 DEFINING RULES .....	10
	2.3.2 INSERTING SOME PATTERNS INTO DATA BASE .....	13
	2.3.3 USE OF IMPORTANT FUNCTIONS .....	14
	2.3.3.1 DB-INSERT .....	14
	2.3.3.2 DB-FIND .....	14
	2.3.3.3 DB-DELETE .....	15
	2.3.3.4 TRYRULE .....	16
	2.3.3.5 RUN .....	17
	2.3.3.6 HALT .....	17
	2.3.4 GENERAL EXECUTION .....	17
	4.3.4 Reliability .....	32
<b>CHAPTER 3</b>		
<b><u>ADVANTAGES OF PRODUCTION RULE SYSTEM</u></b>		24
3.1	FLEXIBILITY .....	24
3.2	EXPANDABILITY .....	25

3.3	EFFICIENCY .....	26
-----	------------------	----

3.4	SIMPLICITY .....	27
-----	------------------	----

5.1	GENERAL .....	36
-----	---------------	----

<b>CHAPTER 4</b>		
<b><u>AASHTO DESIGN PROCEDURE FOR FLEXIBLE PAVEMENTS</u></b> .....		28

<b>CHAPTER 6</b>		
------------------	--	--

4.1	INTRODUCTION .....	28
-----	--------------------	----

4.2	BASICS OF FLEXIBLE PAVEMENTS .....	29
-----	------------------------------------	----

4.3	GENERAL DESIGN VARIABLES .....	30
-----	--------------------------------	----

4.3.1	Performance Criteria /Serviceability .....	30
-------	--	----

4.3.2	Load Condition( $W_{18}$ ) .....	31
-------	----------------------------------	----

4.3.3	Time Constraints .....	31
-------	------------------------	----

4.3.3.1	Performance Period .....	31
---------	--------------------------	----

4.3.3.2	Analysis Period .....	31
---------	-----------------------	----

4.3.3.3	Designed Life .....	31
---------	---------------------	----

4.3.4	Reliability .....	32
-------	-------------------	----

4.3.5	Layer Coefficients ( $a_i$ ) .....	32
-------	------------------------------------	----

4.3.6	Structural Number (SN) .....	34
-------	------------------------------	----

4.3.7	Drainage Coefficients( $m_i$ ) .....	34
-------	--------------------------------------	----



<b>CHAPTER 5</b>	.....	48
<b><u>COMPUTER PROGRAM " BASHROAD " FOR DESIGNING</u></b>		
<b><u>ROADS AND ESTIMATING ITS COST/MATERIALS</u></b>	.....	36
7.9 Layer Coefficient for subbase material ( $a_3$ )	.....	49
5.1 GENERAL	.....	36
7.10 Thickness of the AC layer ( $D_1^*$ )	.....	49
5.2 GENERAL WORKING OF " BASHROAD "	.....	36
7.11 Thickness of the base layer ( $D_2^*$ )	.....	50
<b>CHAPTER 6</b>	.....	51
<b><u>EXECUTION/COMPUTER ANALYSIS - BASHROAD</u></b>	.....	39
7.13 Compacted Volume of Asphalt Concrete Mix	.....	52
7.14 Compacted Volume of Base material	.....	53
6.1 PROCEDURE TO START " BASHROAD "	.....	39
7.15 Compacted Volume of Subbase material	.....	53
6.2 ENTERING INPUT DATA	.....	39
7.16 Weight of Asphalt Concrete Mix	.....	54
7.17 Weight of Base material	.....	55
<b>CHAPTER 7</b>	.....	55
<b><u>DESIGNING PROCEDURE USED IN PROGRAM</u></b>	.....	45
7.19 Cost of Asphalt Concrete Mix	.....	56
7.1 GENERAL	.....	45
7.20 Cost of Base material	.....	56
7.2 Equivalent Single Axle Load (ESAL)	.....	45
7.21 Cost of Subbase material	.....	57
7.3 Load Condition( $W_{18}$ )	.....	46
7.22 Cost of each lane mile	.....	57
7.4 Overall Serviceability Loss ( $\Delta PSI_o$ )	.....	46
7.23 Total cost of the road	.....	58
7.5 Design Serviceability Loss ( $\Delta PSI$ )	.....	46
7.6 Required Structural Number (SN)	.....	47



7.7	Layer Coefficient for AC ( $a_1$ ) .....	48
7.8	Layer Coefficient for granular base material ( $a_2$ ) .....	48
7.9	Layer Coefficient for subbase material ( $a_3$ ) .....	49
8.1	ACCOMPLISHMENTS .....	59
7.10	Thickness of the AC layer ( $D_1^*$ ) .....	49
8.2	RECOMMENDATIONS FOR FUTURE RESEARCH .....	61
7.11	Thickness of the base layer ( $D_2^*$ ) .....	50
7.12	Thickness of the subbase layer ( $D_3^*$ ) .....	51
7.13	Compacted Volume of Asphalt Concrete Mix .....	52
7.14	Compacted Volume of Base material .....	53
7.15	Compacted Volume of SubBase material .....	53
7.16	Weight of Asphalt Concrete Mix .....	54
7.17	Weight of Base material .....	55
7.18	Weight of SubBase material .....	55
7.19	Cost of Asphalt Concrete Mix .....	56
7.20	Cost of Base material .....	56
7.21	Cost of SubBase material .....	57
7.22	Cost of each lane mile .....	57
7.23	Total cost of the road .....	58

## **CHAPTER 8      LIST OF APPENDIX**

### **ACCOMPLISHMENTS AND RECOMMENDATIONS**      59

<b>Appendix-A</b>	<b>PRODUCTION RULES SYSTEM</b>	<b>63</b>
<b>8.1</b>	<b>ACCOMPLISHMENTS</b>	<b>59</b>

<b>Appendix-B</b>	<b>FUNCTION OF PRODUCTION RULE</b>	<b>61</b>
<b>8.2</b>	<b>RECOMMENDATIONS FOR FUTURE RESEARCH</b>	<b>61</b>

<b>Appendix-C</b>	<b>SYSTEM(EXAMPLE)</b>	<b>80</b>
-------------------	------------------------	-----------

<b>Appendix-C</b>	<b>BASHROAD (Programming Code)</b>	<b>85</b>
-------------------	------------------------------------	-----------

<b>Appendix-D</b>	<b>ROAD DESIGNING - EXAMPLE 1</b>	<b>125</b>
-------------------	-----------------------------------	------------

<b>Appendix-E</b>	<b>ROAD DESIGNING - EXAMPLE 2</b>	<b>132</b>
-------------------	-----------------------------------	------------

# **LIST OF APPENDIX**

## **INTRODUCTION**

<b>Appendix-A.</b>	<b>PRODUCTION RULES SYSTEM .....</b>	<b>63</b>
<b>Appendix-B.</b>	<b>FUNCTION OF PRODUCTION RULE -</b>	<b>80</b>
	<b>SYSTEM(EXAMPLE) .....</b>	<b>80</b>
<b>Appendix-C</b>	<b>BASHROAD (Programming Code) .....</b>	<b>85</b>
<b>Appendix-D</b>	<b>ROAD DESIGNING - EXAMPLE 1 .....</b>	<b>125</b>
<b>Appendix-E</b>	<b>ROAD DESIGNING - EXAMPLE 2 .....</b>	<b>132</b>

## **1.2 OVERALL VISUALIZATION OF PROBLEM**

There is a need for developing an expert system to determine the possible alignments and specifications of roads/highways by using a digital map. The ultimate aim of this expert system is to find the best suitable alignment and design a road between any two places such that maximum job can be done just by sitting in front of a computer, and using this system. This expert system should perform the following functions:

- a. Analyze the specified area from a digital map and determine alignments for the required road between any given points in the map, based on the elevation.

## CHAPTER 1

### INTRODUCTION

#### **1.1 GENERAL :**

Due to the large growth rate of traffic in all countries of the world, road planning, alignment, designing and construction is becoming more and more important. The continuous investment required for the nation's transportation infrastructure presents a major and complex task that challenges the ingenuity of every highway administrator and engineer. Selection of suitable road alignment/sites and planning for construction of roads is an expensive and time consuming job. Hence it is important to have a system which can help in performing these jobs at lower cost and in a shorter time so that more money can be used for actual construction of roads.

#### **1.2 OVERALL VISUALIZATION OF PROBLEM**

There is a need for developing an expert system to determine the possible alignments and specifications of roads/highways by using a digital map. The ultimate aim of this expert system is to find the best suitable alignment and design a road between any two places such that maximum job can be done just by sitting in front of a computer, and using this system. This expert system should perform the following functions:-

- a. Analyze the specified area from a digital map to give possible alignments for the required road between any given points by keeping in view the elevation/



slopes, barriers like steep hills, buildup/urban areas, rivers/streams, etc. This can be done in two ways :-

- (1) Click two points(with the help of cursor), which are required to be connected by road, on the map(displayed on the screen) and apply

Artificial Intelligence search techniques in order to find different suitable alignments by taking into account any barriers/constraints.

- (2) Mark different alignments between two points with the help of cursor.

In this approach, Artificial Intelligence search techniques are not required to find different suitable alignments.

- b. Design the road( i.e. determining thickness of different pavement layers) by keeping in view the soil bearing capacity/road bed soil resilience modulus and

other available data. Required data can be obtained in the following ways :-

- (1) Part of the data may be provided by the engineer, like number of lanes required, structural number required to protect Base layer, SubBase layer and Road bed soil(structural number can also be calculated if not provided by the engineer), material properties(i.e. Modulus of

### 1.3 RECOMMENDATIONS

As numerous coefficients for Base and SubBase layers, reliability, compacted densities required for Asphalt concrete mix, Base material and SubBase material, expected traffic, etc. ( parameters are capitalized for emphasis).

- (2) Some data can also be obtained from the digital map i.e. length of road required to be constructed for different alignments, ground slope/



Language, which elevations, under ground water table, barriers, road bed soil resilience modulus etc. Some additional coverage's (i.e. road bed soil resilience

a. Design modulus-coverage etc.) may have to be digitized for this purpose in order to because they are not digitized/available in usual practice. system

c. Calculate the earth work required, the quantity of materials required for base (SP) layer, sub base layer and surfacing. Earth work can be calculated by using ground slope/elevations from digital map. ing thickness of different pavement

d. Cater for any additional requirement, like a bridge over a water stream, culverts, etc. nation/data extracted from digital map.

e. Estimate cost of materials required for Asphalt concrete layer, Base layer and SubBase layer of road for all possible alignments/roads. as/roads.

f. Estimate total cost of all possible alignments/roads. they are constructed on

g. Taking into account the lengths, cost, materials and the effort required for different possible alignments/roads, the system should decide about the best option available for making this road.

(g) Comparing lengths of all roads appearing that they are

### 1.3 **RECOMMENDED/FOLLOWED APPROACH :**

As numerous requirements and problems have to be dealt with, so there should be a versatile, flexible, expandable and comprehensive system in order to fulfill these requirements. As Artificial Intelligence has the capabilities to tackle such problems in an appropriate way, it was decided to design this system in List Programming (LISP)

constraints/political, economic, social, etc.

Language, which is an Artificial Intelligence Language. Brief description of the approach is as under :-

- a. Designing of a production rule system in List Programming(LISP) Language in order to provide the flexibility, expandability, and versatility to the system .
- b. Developing a system in the production rule system, in List Programming(LISP) language to include the following :-

- (1) Designing of roads(i.e. determining thickness of different pavement layers) with the help of available data (provided by user) and information/data extracted from digital map.

- (2) Estimation of material/effort for Asphalt concrete layer, Base layer and SubBase layer of road for all possible alignments/roads.

- (3) Cost estimation of all roads assuming that they are constructed on different possible alignments.

- (4) Assigning priorities to construction of the road on different possible alignments by :-

- (a) Comparing lengths of all roads assuming that they are constructed on different alignments.

- (b) Comparing materials and the effort required for different roads assuming that they are constructed on different alignments.

- (c) Comparing cost of all roads for different alignments.

- (5) Decide about the best option by taking into account the priorities, constraints(political, ecological etc.), barriers etc.

## 1.4 **PROBLEMS ADDRESSED :**

This document addresses road(flexible pavement) designing, material estimation, and cost estimation. A system is designed which contains programs to deal with these problems. So far it is not connected with a digital map i.e. it can only accept data from the user but not from a digital map. Following problems are addressed and solved :-

### 1.4.1 **Designing of a production rule system :**

Although designing of a production rule system itself is a major task and requires lot of effort and time, but because of its numerous advantages(described later in this document) we can say that a system, which is made by just defining different functions, can never give performance comparable to that of a system which is designed with the help of a production rule system.

As this system is designed as a part of an overall system, so a production rule system was considered essential and is designed (using Artificial Intelligence) in List Programming (LISP) Language in order to provide the flexibility, expandability, and versatility to the system . This production rule system was checked and run very extensively and hopefully most of the errors/problems have been solved which could create trouble for future changes/expansion. Some advantages and brief description of the production rule system is given later.

A number of rules are made for designing. Following are the outputs available for the Engineer/ Users :-

- a. Structural Number to provide the base for the road.
- b. Structural Number to provide the base for the road.

#### 1.4.2 **Designing of Roads** : number to protect the roadbed soil (SN3)

For designing different layers of flexible pavement, AASHTO(American Association of State, Highway, and Transportation Officials) procedure for flexible pavement thickness design is followed. A number of functions are defined to get required data input from the user. Following data is obtained from the

Engineer/ Users :- of the base layer (D2\*)

a. Number of lanes required and width of each lane. (D3\*)

b. Length of Road .

#### 1.4.3 **Material** d. Material Properties (Modulus of resilience) for Asphalt concrete, Base

and SubBase layers .ity of materials in different layers of road , a

number e. Design period. defined to get required data input from the user.

Follow f. Traffic. obtained from the Engineer/User :-

g. Reliability. density of Asphalt Concrete layer

h. Overall standard deviation (that accounts for standard variation in

e. materials and construction, the probable variation in the traffic

Different prediction, and the normal variation in pavement performance for a sets of

road. Follow given design traffic application). User (details would be given later ) :-

i. Serviceability loss. volume of Asphalt Concrete layer

j. Drainage coefficients for Base and SubBase layers.

A number of rules are made for designing different layers of road . Following outputs are available for the Engineer/User (details would be given later):-

a. Structural Number to protect the base layer (SN1)

b. Structural Number to protect the SubBase layer (SN2)



- 1.4.4 **Costs**
- c. Structural Number to protect the roadbed soil (SN3)
  - d. The value of layer coefficient for AC (a1)
  - e. The value of layer coefficient for base (a2)
  - f. The value of layer coefficient for subbase (a3)
  - g. Thickness of the AC layer (D1\*)
  - h. Thickness of the base layer (D2\*)
  - i. Thickness of the subbase layer (D3\*)

For estimation of cost of materials in different layers of road, different rules

1.4.3 **Material estimation** : Inputs are available for the Engineer/User :-

For estimation of quantity of materials in different layers of road, a number of functions are defined to get required data input from the user.

Following data is obtained from the Engineer/User :-

- a. Compacted density of Asphalt Concrete layer.
- b. Compacted density of Base layer.
- c. Compacted density of SubBase layer.

Different rules are made for estimation of quantity of materials in different layers of road. Following outputs are available for the User (Details would be given later) :-

- a. Total compacted volume of Asphalt Concrete mix .
- b. Total compacted volume of Base material .
- c. Total compacted volume of SubBase material .
- d. Total weight of Asphalt Concrete mix .
- b. Total weight of Base material .
- c. Total weight of SubBase material .



#### 1.4.4 Cost estimation :

Similarly for estimation of cost of materials in different layers of road, different functions are defined to get following data input from the user :-

2.1 **GENERAL** :

- a. Cost of Asphalt Concrete mix in Dollars /ton .

- b. Cost of Base material in Dollars /ton .

- c. Cost of SubBase material in Dollars /ton .

- d. Miscellaneous overhead charges in Dollars/Lane-mile .

For estimation of cost of materials in different layers of road , different rules are made. Following outputs are available for the Engineer/User :-

- a. Total cost of Asphalt Concrete layer material .

- b. Total cost of Base layer material .

- c. Total cost of SubBase layer material .

- d. Cost of road for each lane mile .

- e. Total cost of the road .

```
(rule priority rulename pattern (var1 (var2 ... (varN) argument ->)
(S-exp1 S-exp2 ..... S-expN))
```

A brief description of each component of the structure used to store the rule is as under:-

2.2.1 **priority** . It is a number between 0 - 655 (inclusive) . It gives the sequence in which the rules would be fired from the Production memory, starting from highest priority rules (oldest of them) . If any rule fires, it would start from the beginning again i.e. from highest priority rules.

2.2.2 **rule-name** . It is a symbol to be assigned to the rule.

2.2.3 **pattern** . It is a separator in rule , and is represented as

2.2.4 **s-exp** . It is a list of **CHAPTER 2** executed when rule fires.

2.2.5 **pattern** . It is logical term containing different variables like (lvar1 lvar2 ..... lvarN). A rule fires if the pattern matches i.e. if all the entries/

## **PRODUCTION RULE SYSTEM**

### **2.1 GENERAL :**

A comprehensive production rule system is designed in order to provide the flexibility, expandability, efficiency and simplicity of programming . It consists of a number of functions, interrelated in a complex manner, however a simple and brief description of some important terms is given below. (complete code is attached as Appendix-A, page 62-78)

### **2.3 FUNCTION OF PRODUCTION RULE SYSTEM :-**

### **2.2 STRUCTURE TO STORE THE RULE :**

The structure used to store the rule is a list of rule-name, pattern, list of variables in pattern and the list of S-expressions :-

**(rule priority rulename pattern(lvar1 lvar2 ...lvarN) arrow(-->)**  
**(S-exp1 S-exp2 ..... S-expN) )**

#### **2.3.1 DEFINING RULES :**

A brief description of each component of the structure used to store the rule is as under:-

**2.2.1 priority** . It is a number between 0 - 400 (inclusive) . It gives the sequence in which the rules would be fired from the Production memory, starting from highest priority rules (oldest of them). If any rule fires, it would start from the beginning again i.e. from highest priority rules.

**2.2.2 rule-name** It is a symbol to be assigned to the structure.

**2.2.3 arrow** . It is a separator in rule , and is represented as “ --> ”

**2.2.4 s-exp** . It is a list of s-expressions to be executed when rule fires.

**2.2.5 pattern** . It is a logical term containing different variables like (!var1 !var2 ..... !varN). A rule fires if the pattern matches i.e. if all the entries/variables of the inserted pattern matches with the pattern of a rule in the production memory. An entry of the inserted pattern, attached with “ ! ” sign will match anyway with the corresponding entry of the pattern of a rule (in the production memory) irrespective of the type/number of characters.

((minusp diff)(setq sn1 (+ sn1 0.1)))

## 2.3 FUNCTION OF PRODUCTION RULE SYSTEM :-

The overall functioning of the production rule system is elaborated with the help of an example as shown in Appendix-B, page 79-83). Detail is shown below:-

Structural Number to protect the Bone layer = SN1 = -d\* sn1 )  
(setq (get 'sn1 'value) sn1)

### EXAMPLE

#### 2.3.1 DEFINING RULES :

Any number of rules (for this system, up to 400 rules) can be defined by following the structure,

```
(rule priority rulename (!var1 !var2 ...!varN) -->
  ( S-exp1 )
  (terpri)(terpri)
  ( S-exp2 )
  ( ..... )
  ( S-expN ) )
```

(prin1 a1)

(setq (get 'a1 'value) a1)

Let there be three rules stored in Production memory as :-

(db-delete '(calculate a1 !j)) (1)

(rule 280 calculate-SN1 (calculate sn1 !j) -->

(rule 200 calculate-D2\* (calculate D2\* !j) -->

(setq diff (- (+ (- (+ (\* Zr S0) (\* 9.36 (log (+ SN1 1) 10)))) 8.27)

(\* 2.32 (log Mr-base 10)))/(log (/ Delta-PSI 2.7) 10)

(setq D2\* (- (get 'a2 'value) SN1\*) (\* a2 (get 'm2 'value))))

(+ 0.4 (/ 1094 (expt (+ SN1 1) 5.19)))) (log W18 10)))

(setq D2\* (round D2\*))

(cond

(setq SN2\* (\* a2 D2\* m2))

((minusp diff)(setq sn1 (+ sn1 0.1)))

(else

(> sn1 14) (db-delete '(calculate sn1 !j))

((<= D2\* 0)(terpri)(terpri)(format t"

(format t" Please check your data for errors"))

The Base layer is not required.")(setq D2\* 0)

(t (format t"

(db-insert '(calculate D3\* 'value))(db-delete '(calculate D2\* !j)))

Structural Number to protect the Base layer = SN1 = ~d" sn1 )

(setf(get 'sn1 'value) sn1) SN2\*(terpri)(terpri)(format t"

(setq sn2 1) and thickness of the Base layer = D2 = ~d inches" D2)

(db-insert '(calculate sn2 value ))

(db-delete '(calculate sn1 !j)))) Base layer = D2 = ~d inches " D2")

(terpri)(terpri)(format t"

(2)

(rule 240 calculate-a1 (calculate a1 !j) -->

(terpri)(terpri)

(setq a1 (+ (\* 0.4 (log (/ (get 'e-ac 'value ) 435000) 10)) 0.44))

(print" The value of layer coefficient for AC =a1 = ")

(prinl a1)

(setf (get 'a1 'value)a1)

(terpri)(terpri)



```
(self (get 'D2* 'value) D2*))
(db-insert '(calculate a2 value ))
(db-insert '(calculate D3* value ))
(db-delete '(calculate a1 !j)))
(db-delete '(calculate D2* !j)))
```

(3)

```
(rule 200 calculate-D2* (calculate D2* !j) -->= ))
```

```
(terpri)(terpri)
(delete '(calculate D2* !j))))))
```

```
(setq D2 (/ (- (get 'sn2 'value) SN1*) (* a2 (get 'm2 'value))))
```

### 2.3.2 INSERTING DATA INTO DATA BASE :

```
(setq D2* (round D2))
```

```
(setq SN2* (* a2 D2* m2))
```

some p (cond into data base with the help of a function " db-insert " :-

```
USER: ((<= D2* 0)(terpri)(terpri)(format t"
```

```
(The Base layer is not required. ")(setq D2* 0)
```

```
USER(2) (db-insert '(calculate D3* value ))(db-delete '(calculate D2* !j)))
```

```
((>= (+ SN1* SN2*) SN2)(terpri)(terpri)(format t"
```

```
US Calculated thickness of the Base layer = D2 = ~d inches" D2)
```

```
(CALC (terpri)(format t"
```

```
US Rounded thickness of the Base layer = D2* = ~d inches " D2*)
```

```
(VOL (terpri)(terpri)(terpri)(format t"
```

US The actual combined Structural Number of the Surface layer and Base

course is greater than the Structural Number of the Sub-Base course i.e.

$(SN1* + SN2*) \geq SN2$  Or Numerically ,

Hence following pattern  $(\sim d + \sim d)(\sim d) \geq \sim d \implies OK$  "

```
( (calculate D1 SN1* SN2* (+ SN1* SN2*) SN2 )
```

```
(terpri)(terpri)
```

```
(calculate SN1 value )
(setf (get 'D2* 'value)D2*)
(calculate a1 value )
(db-insert '(calculate D3* value ))
(volume AC layer)
(db-delete '(calculate D2* !j)))
(volume Base layer)
(t(and(db-insert '(re-calculate D2* value ))
(db-delete '(calculate D2* !j)))))
```

### 2.3.3 USE OF IMPORTANT FUNCTIONS

#### 2.3.2 INSERTING SOME PATTERNS INTO DATA BASE :

Suppose that the following commands are given at LISP prompt to insert some patterns into data base with the help of a function “ db-insert ” :-

```
USER(1): (db-insert '(calculate D1* value ))
```

```
(CALCULATE D1* VALUE)
```

```
USER(2): (db-insert '(calculate SN1 value ))
```

```
(CALCULATE SN1 VALUE)
```

```
USER(3): (db-insert '(calculate a1 value ))
```

```
(CALCULATE A1 VALUE)
```

```
USER(4): (db-insert '(volume AC layer))
```

```
(VOLUME AC LAYER)
```

```
USER(5): (db-insert '(volume Base layer))
```

```
(VOLUME BASE LAYER)
```

Hence following patterns are inserted into data base:-

```
( (calculate D1* value )
```

(calculate SN1 value ) the pattern (calculate D1\* !j ) is present in the  
 (calculate a1 value ) calculate and D1\* were matched so it unified " !j "  
 (volume AC layer)  
 (volume Base layer) )

### 2.3.3.3 DB-DELETE :

### 2.3.3 USE OF IMPORTANT FUNCTIONS : (term) or (db-delete '(var1 var2.....varN))

is used to delete a pattern from the data base. Here at least one variable in

#### 2.3.3.1 DB-INSERT : should have " ! " sign to allow unification. Any logical term

can be The function (db-insert pattern) or (db-insert '(var1 var2 .....varN ) ) is used to insert a pattern into data base. Any logical term can be inserted into indexed data base on requirement bases. Performance is shown above.

This shows that the pattern (volume Base layer) was present in the

#### 2.3.3.2 DB-FIND : which matched with the pattern (volume Base !j), so it was

deleted The function (db-find pattern) or (db-find '(var1 var2 .....varN ) )

is used to find a pattern from the data base. Here at least one variable in

the pattern should have " ! " sign to allow unification (variable having

" ! " sign can have unification with any corresponding term in the

pattern(present in the data base) . Performance at LISP prompt is as under,

```
USER(5) : (db-find '(calculate D1* !j ))
```

```
((((( !j VALUE ))))
```

**2.3.3.4 IRYE** This shows that the pattern (calculate D1\* !j ) is present in the data base. The terms calculate and D1\* were matched so it unified “!j” and “value”. “rulename”. It tries to match the pattern of rule, rule having name “rulename” with the inserted patterns. If any pattern

**2.3.3.3 DB-DELETE :** It will be fired (all the s-expressions will be executed). If no pattern The function (db-delete pattern) or (db-delete '(var1 var2.....varN)) is used to delete a pattern from the data base. Here at least one variable in the pattern should have “!” sign to allow unification. Any logical term can be deleted from indexed data base on requirement bases. Performance is as under, is an assumed value of a1)

```
USER(6) : (db-delete '(volume Base !j))
expressions of this rule are executed. Here the ((VOLUME BASE LAYER)) deleted with the help
```

This shows that the pattern (volume Base layer) was present in the data base which matched with the pattern (volume Base !j), so it was deleted. Following patterns are present in data base:-

Now following patterns are remaining in data base:-

```
( (calculate D1* value )
(calculate SN1 value )
(calculate a1 value )
We can (volume AC layer) )
```

```
USER(8) : (db-delete '(calculate a1 value))
expressions of this rule are executed. Here the ((CALCULATE A1 VALUE)) deleted with the help of condition.
USER(9) : (db-insert '(calculate a1 value))
```



### 2.3.3.4 TRYRULE : (CALCULATE A1 VALUE)

Again The function (tryrule rulename) is used to fire/execute only one rule having name " rulename ". It tries to match the pattern of rule, rule having name " rulename " with the inserted patterns. if any pattern matches, this rule will be fired(all the s-expressions will be executed). If no pattern matches, it will exit with nil as output. If we give following command at the LISP prompt,

2.3.3.5 RUN USER(7): (tryrule calculate-a1)

THE VALUE OF LAYER COEFFICIENT FOR AC =  $A1 = 0.66$

(Here 0.66 is an assumed value of a1)

The rule "calculate-a1" is fired and all s-expressions of this rule are executed. Here the pattern "(calculate a1 !j)" is deleted with the help of command (db-delete '(calculate a1 !j)) and another pattern "(calculate a2 value)" is inserted into data base with the help of function "db-insert "

2.3.3.6 Now following patterns are present in data base:-

( (calculate D1\* value )  
(calculate SN1 value )  
(volume AC layer)  
(calculate a2 value ) )

We can again bring the data base in same old condition by,

2.3.4 GENERATE USER(8): (db-delete '(calculate a2 !j))

In this example((CALCULATE A2 VALUE))

USER(9): (db-insert '(calculate a1 value ))

Now first of all (CALCULATE A1 VALUE) is a rule having top priority. Again following patterns are present inserted in data base:-

"calcul (is-SN (calculate D1\* value ) calculate SN1 lj) "

In addition to the above different functions, following steps would be executed :- (volume AC layer)

a. Check, (calculate a1 value ) is on "SYNTAX-BAD ", that the syntax of rule macro is not wrong.

2.3.3.5 **RUN** : with the help of a function "IS-VAR ", if the term is a variable

b. with The function "(RUN)" tries to match all pattern of rules (stored in Production memory), starting from highest priority rules (oldest of them) with the inserted patterns. If any pattern matches, that rule will be fired( all s-expressions will be executed), and it will start from the beginning. If no pattern matches, it will exit with nil as output.

c. Check, with the help of a function "Rest-formed", if the term after the

2.3.3.6 **HALT** : is OK.

f. Pattern When the rules are being fired with the help of "(RUN)", this function "(HALT)" can be used to exit from the system to LISP prompt i.e. it will stop executing "RUN". One "HALT" is required to stop execution of each "RUN".

2.3.4 **GENERAL EXECUTION** : any of the above conditions is not match, an error

In this example let us call the function "RUN" at the LISP prompt,

USER(10) : (RUN)

Now first of all it will try to match the pattern of rule having top priority(280) and rulename "calculate-SN1"<sup>2</sup> with the inserted patterns. The rule "calculate-SN1" has the pattern "(calculate SN1 !j)". It will try to match the pattern

In addition to a number of different functions, following steps would be executed :- "calculate-SN1" with the inserted patterns. For this purpose it will take

- a. Check, with the help of a function "SYNTAX-BAD", that the syntax of rule macro is not wrong. "calculate", of first inserted pattern "(calculate D1"
- b. Check, with the help of a function "IS-VAR", if the term is a variable with "!" sign in front. "!", of the pattern "(calculate SN1 !j)" of same
- c. Check, with the help of a function "WELL-FORMED", if the term is well formed or not. "ue)". This match is not successful as the terms are
- d. Check, with the help of a function "function-OK", if the terms are rule variables or numbers or lists. "n matching occurred."
- e. Check, with the help of a function "Rest-formed", if the term after the of function is OK. "match with the first term, "calculate", of second
- f. Pattern matching is done by using some functions, mainly "UNIFY" (Which takes two logical terms and checks if they are well formed, & function is a symbol.), and "MATCH" (Which checks if term1 is a variable, then matches the variable by calling another function "MATCH-VAR". If term2 is a variable, then calls "MATCH-VAR" with term2 as var.). If any of the above conditions do not match, an error message is printed. There are some other conditions also, which are not

rule and match with the third term, "some" of variable function pattern

mentioned here to avoid complexity. However they can be seen from the code, attached as Appendix-A, page 62-78, if required.

- g. When all the conditions are satisfied, then it will try to match the pattern “(calculate SN1 !j)” of rule having top priority(280) and rulename “calculate-SN1” with the inserted patterns. For this purpose it will take first term, “calculate”, of the pattern “(calculate SN1 !j)” and match with the first term, “calculate”, of first inserted pattern “(calculate D1\* value)”. This match is successful as both the terms are same.
- h. Take second term, “SN1”, of the pattern “(calculate SN1 !j)” of same rule, and match with the second term, “D1\*”, of first inserted pattern “(calculate D1\* value)”. This match is not successful as the terms are different. This shows that first inserted pattern is unable to fire the rule “calculate-SN1” as no pattern matching occurred.
- i. Now take first term, “calculate”, of the pattern “(calculate SN1 !j)” of same rule, and match with the first term, “calculate”, of second inserted pattern “(calculate SN1 value)”. This match is successful as both the terms are same.
- j. Again take second term, “SN1”, of the pattern “(calculate SN1 !j)” of same rule, and match with the second term, “SN1”, of second inserted pattern “(calculate SN1 value)”. Again This match is successful as both the terms are same.
- k. Now take third term, “!j”, of the pattern “(calculate SN1 !j)” of same rule, and match with the third term, “value”, of second inserted pattern



- m. “(calculate SN1 value ) ”. This match is anyway successful as one terming has “ ! ” sign. Thus the rule “calculate-SN1” will be fired and all s-expressions of this rule will be executed. Output on the screen would be :  $280$  and  $240$ , (i.e. the next priority is  $240$ ) so it will try to match the pattern Structural Number to protect the Base layerity =  $SN1 = x$  rulename “calculate-a1” with  $x$  Where  $x$  = calculated SN value
- n. Note that this pattern is deleted by the command (db-delete '(calculate SN1 !j)). It is done in order to avoid continuous unlimited firing of the same rule. Also another pattern “(calculate sn2 value )” is inserted into data base with the help of function “ db-insert ” ( supposing that the first two conditions in this rule i.e. “(minusp diff)” and “(> SN1 14)” are not true ).
- o. Now following patterns are inserted/present in data base:-  
 ( calculate D1\* value )  
 (volume AC layer)  
 (calculate a1 value )  
 (calculate sn2 value )
- l. As one of the rules is fired, now it will restart again from the beginning and try to match the pattern “ (calculate SN1 !j) ” of rule having top priority(280) and rulename “calculate-SN1” with the inserted patterns. If this pattern was not deleted from the data base, the same rule would continuously keep firing until it is deleted from the data base. Now there would be no match as no such pattern is present in the data base.

- m. After finding no match of the pattern “(calculate SN1 !j)” of rule having top priority(280), it will keep decreasing the priority by one and will check for a rule with a priority lower than 280. As there is no rule having a priority within 280 and 240, (i.e. the next priority is 240) so it will try to match the pattern “(calculate a1 !j)” of rule having priority(240) and rulename “calculate-a1” with the inserted pattern.
- n. Here first it will try to match the pattern “(calculate a1 !j)” of rule having priority(240), with the first inserted pattern “(calculate D1\* value)”. This would fail as the second terms of both patterns do not match. So it will try to match the pattern “(calculate a1 !j)” of same rule with the second inserted pattern “(volume AC layer)”. This would again fail as
- q. the first terms do not match.
- o. Now it will try to match the pattern “(calculate a1 !j)” of same rule with the third inserted pattern “(calculate a1 value)”. This match would be successful as both the patterns are same. Thus the rule “calculate-a1” will be fired and all s-expressions of this rule will be executed. Output on the screen would be :
- r. Again after find The value of layer coefficient for AC = a1 = x  
 Having priority 240, it will keep decreasing. Where x = calculated a1 value.  
 Here the pattern “(calculate a1 !j)” is deleted with the help of command (db-delete '(calculate a1 !j)) in order to avoid continuous unlimited firing of the same rule. Also another pattern “(calculate a2 value)” is inserted

into data base with the help of function is “db-insert”.

Now following patterns are present in data base:-

- s. ( (calculate D1\* value ) match of the pattern “ (calculate D2\* !j) ” of rule (volume AC layer) it will keep decreasing the priority by one and will (calculate sn2 value ) a priority lower then 200. There is no rule pre (calculate a2 value ) ) memory with a priority less than 300, so it would
- p. Because again one of the rules was fired, so it will restart from the beginning and try to match the pattern “ (calculate SN1 !j) ” of rule having top priority(280) and rulename “calculate-SN1” with the inserted patterns. There would be no match as no such pattern is present in the data base.
- q. After finding no match of the pattern “ (calculate SN1 !j) ” of rule having top priority (i.e. 280), it will keep decreasing the priority by one and will check for a rule with a priority lower then 280. The next priority is 240, so it will try to match the pattern “ (calculate a1 !j) ” of rule having priority(240) and rulename “calculate-a1” with the inserted patterns. There would be no match as no such pattern is present in the data base.
- r. Again after finding no match of the pattern “ (calculate a1 !j) ” of rule having priority 240, it will keep decreasing the priority by one and will check for a rule with a priority lower then 240. The next priority is 200, so it will try to match the pattern “ (calculate D2\* !j) ” of rule having priority(200) and rulename “calculate-D2\*” with the inserted patterns.

There would be no match again as no such pattern is present in the data base.

### ADVANTAGES OF PRODUCTION RULE SYSTEM

- s. Similarly after finding no match of the pattern “(calculate D2\* !j)” of rule having priority 200, it will keep decreasing the priority by one and

#### 3.1 FLEXIBILITY :

The production rule system is designed in order to provide the flexibility to the programmer in following ways:-  
end.

- a. A programmer can store the rules in the production memory by giving the priorities in descending order in which he wants the rules to fire.
- b. If at any stage, the programmer wants to change the order in which the rules should fire, he can do so just by changing their priorities.
- c. A programmer has the option to let the rules fire one by one, by inserting only one pattern each time, OR he can fire a sequence of rules, one after the other by inserting that many patterns at one time.
- d. If the programmer wants to skip some rules, which are not to be fired for a particular application, he can do so just by skipping the patterns which are required to fire those rules.
- e. A programmer can start the application at any time during the program just by inserting the pattern(s) which he/she wants to fire. The program will be started by giving the command “RUN”.
- f. In the same manner the programmer can stop the application at any time during the program just by giving the command “STOP”.



g. A programmer has the **CHAPTER 3** fire one after the other in a chain

he inserting the pattern (which is required to fire the desired rule) during

the execution of the s-expressions of each (current) rule.

changed, the rules and their

### 3.1 **FLEXIBILITY :**

#### 3.2 **EXPANDABILITY :**

The production rule system is designed in order to provide the flexibility to the

The production rule system provides ability to programmer to expand the program

programmer in following ways:-

as per his requirements in the following ways:-

a. A programmer can store the rules in the production memory by giving the

a. A programmer can add any number of rules in the production memory at any priorities in descending order in which he wants the rules to fire.

b. If at any stage, the programmer wants to change the order in which the rules wants the rules to fire. The rules are given the priority by leaving some gap should fire, he can do so just by changing their priorities.

c. A programmer has the option to let the rules fire one by one, by inserting only Nine new rules can be added between two rules having successive one pattern each time , OR he can fire a number of rules, one after the other by

inserting that many patterns at one time.

b. If at any stage, the programmer has a pattern which is not in the nine rules d. If the programmer wants to skip some rules, which are not to be fired for a between one pair of rules having successive priorities, he can do so just by particular application, he can do so just by not inserting the patterns which are changing their priorities and inserting the patterns which are required to fire those rules.

e. A programmer can start the application at any time during the program just by having successive priorities of

inserting the pattern(s) which is/are required to fire the desired rule(s) and by

c. Presently a total of 400 rules are in the production memory having giving the command "RUN" .

priorities from 400 to 1 . If a programmer wants to change

f. In the same manner the programmer can stop the application at any time 400 rules are needed for the

during the program just by giving the command "HALT" .

the upper limit of priority - that is, 400 - and the lower limit of priority - that is, 1 -

- g. A programmer has the option to let the rules fire one after the other in a chain by inserting the pattern (which is required to fire the next desired rule ) during the execution of the s-expressions of each (current) rule.

### 3.2 **EXPANDABILITY** :

The production rule system provides ability to programmer to expand the program as per his requirements in the following ways:-

- a. A programmer can add any number of rules in the production memory at any time by giving the priorities in descending order (from 400 to 1) in which he wants the rules to fire. The rules are given the priorities by leaving some gap between each pair of rules having successive priorities (i.e. 290 280 etc.). Nine new rules can be added between each pair of rules having successive priorities.
- b. If at any stage, the programmer feels a need to add more than nine rules between one pair of rules having successive priorities, he can do so just by changing their priorities and making more room. i.e. change priority 290 to 295 and 280 to 275. Now 19 new rules can be added between the pair of rules having successive priorities of 295 and 275.
- c. Presently a total of 400 rules can be added in the production memory having priorities from 400 to 1. If a programmer feels at any stage that more than 400 rules are needed for the system, he can create more room just by changing the upper limit of priority (i.e. 400) to any desired number. i.e. change upper

limit of priority from 400 to 800. This would create a room for a total of 800 rules to be added in the production memory having priorities from 800 to 1.

- d. For future changes/expansion, it is possible for a programmer (even in the

3.4 **3.1** absence of originator of the code ) to easily understand the rules and their working without going into the details of complex programming/code of the and easy to production rule system itself. This is made possible because of the following :-

- a. The programmer can define the rules and can give them priority, in which he

3.3 **EFFICIENCY :** is to fire after giving a deliberate consideration to the overall aim

The production rule system is designed such that the programmer can get better efficiency with less effort. This is made possible because of the following :- the order of

- a. A programmer can give any priority to a rule in which he wants the rules to fire, running more easy, simple and fast.

- b. To change the order of rule firing, he just needs to change their priorities. This makes the programming more efficient, convenient and fast.

- c. During pattern matching, first term of the pattern of a rule is considered as key. The second term is only tried when first term is matched otherwise

- d. pattern of next rule is taken for pattern matching. In this way, effort is not wasted in matching all the patterns.

- d. Efficiency is also achieved by having a minimum number of patterns in the data base at any time. In this way pattern of each rule (being tried at its priority) is matched with only that many patterns which are present in the data base at that time. This technique also makes handling of program easy and

simple as the programmer has to deal with minimum number of patterns at one time.

## AASHTO DESIGN PROCEDURE FOR FLEXIBLE PAVEMENTS

### 3.4 SIMPLICITY :

One of the aims of a production rule system is to make the programming simple and easy to handle for the programmer. This is made possible because of the following :-

- a. The programmer can define the rules and can give them priority, in which he wants the rules to fire after giving a deliberate consideration to the overall aim of programming.
- b. Afterwards if at any stage, the programmer feels a need to change the order of rule firing, he just needs to change their priorities. This makes the programming more easy, simple and fast.
- c. The programmer has the option to insert as many patterns at one time as he can handle easily. He can let the rules fire one by one, by inserting only one pattern each time, OR can fire a number of rules, one after the other by inserting that many patterns at one time.
- d. Skipping of some rules, which are not to be fired for a particular application, is possible just by not inserting the patterns which are required to fire those rules.
- e. The production rule system makes it simple to change/expand the program in future because it is easy to understand the rules and their working without going into the details of the production rule system itself.



design procedure incorporates the use of resilience modulus testing for materials characterizations and determination of layer coefficients, direct consideration of the effect of drainage on material performance, and a rational approach to adjusting designs to account for environmental considerations.

## CHAPTER 4

### **AASHTO DESIGN PROCEDURE FOR FLEXIBLE PAVEMENTS**

#### **4.1 INTRODUCTION :**

The AASHTO design procedure for flexible pavements is based on results obtained from the AASHO Road Test conducted in the late 1950's and early 1960's in North Illinois. The AASHO Road Test identified many important pavement design concepts, including the demonstration of major influences of traffic loads and repetition upon design thickness. Equally important was the development of serviceability-performance method of analysis, which provided a quantifiable way of defining failure conditions based on user-oriented definition rather than one based on structural failure.

(PSI). It is an empirical procedure, derived from experience or observation and is generally used to determine required pavement thickness, the number of load application to failure, or the occurrence of distresses as a function of pavement materials properties (subgrade type, climate, traffic, etc.). The advantage to use empirical models is that they tend to be simple and easy to use but they are usually only accurate for the exact conditions under which they are developed. However the effect of this disadvantage was reduced considerably by generalizing the design equations to make them applicable to broader set of design variables. The original flexible pavement design equations were issued in 1961, but over the years, several modifications and improvements have been made to the procedures, with the significant changes occurring in 1986. The current

design procedure incorporates the concepts of design reliability and variability of design inputs, the use of resilience modulus testing for materials characterizations and determination of layer coefficients, direct consideration of the effect of drainage on material performance, and a rational approach to adjusting designs to account for environmental considerations.

The AASHTO design procedure is based on providing enough strength in the pavement layers to prevent overloading of roadbed soil by the applied loads. The principal objectives are to avoid permanent deformation in the lower pavement layers that cause rutting and critical radial stresses in the upper layers which cause alligator or fatigue cracking. The most effective way to reduce subgrade vertical compression stresses is usually to thicken the entire pavement structure. Reduction of radial strain in the upper layers can be accomplished by reducing stiffness of the layers or increasing their thickness. The pavement performance is measured by a Present Serviceability Index (PSI), which is a mean slope variance in the two wheel paths, the amount of cracking and patching in the pavement surface, and the depth of rutting in the wheel paths.

AASHTO Road Test were 4.2 for flexible pavement and 4.3 for rigid pavement.

#### **4.2 BASICS OF FLEXIBLE PAVEMENTS :** (PSI) is the lowest index that will be

Flexible pavement structure may consist of three layers, designated the sub-base course, base course and surface course [1]. The surface course consists of mixture of mineral aggregates and bituminous materials. It must be designed to resist the abrasive force of traffic, and to provide a smooth and uniform riding surface. The base course usually consists of aggregates, such as crushed stones, crushed slag etc. It performs its

major function as a structural portion of the pavement. The sub-base course usually consists of a compacted layer of granular material [2].

#### **4.3 GENERAL DESIGN VARIABLES :**

Definitions of some of the terms/variables, used in designing of flexible pavements are given below:-

##### **4.3.1 Performance Criteria /Serviceability :**

Serviceability of a pavement is defined as its ability to serve the type of traffic for which it was designed. Primary measure of serviceability used by AASHTO procedures is Present Serviceability Index (PSI), which ranges from a minimum of 0 (representing impassable conditions) to a maximum of 5 (representing perfect conditions). Initial and terminal pavement serviceability indexes must be established to compute the total change in serviceability that will be used in the design equation. The initial pavement serviceability index ( $PSI_I$ ) is a function of pavement design and construction quality. Typical values from the AASHTO Road Test were 4.2 for flexible pavement and 4.5 for rigid pavement. The terminal pavement serviceability index ( $PSI_T$ ) is the lowest index that will be tolerated before rehabilitation, resurfacing or reconstruction becomes necessary, and it generally varies with the importance or functional classification of the pavement. Recommended terminal serviceability indexes are 2.5 or higher for major highways and 2.0 to 2.5 for less important pavements.

#### 4.3.2 Load Condition( $W_{18}$ ) :

The AASHTO procedures are based on  $W_{18}$ , which is a proxy of prevailing loading conditions for the pavement to be designed and it represents the total number of equivalent 18-kip single axle load applications during the analysis period.

#### 4.3.3 Time Constraints :

**4.3.3.1 Performance Period :** It is the period of time that elapses as a new or rehabilitated pavement structure deteriorates from its initial stage of the serviceability to its terminal serviceability and requires the designed reliability of rehabilitation.

**4.3.3.2 Analysis Period :** It is the period of time that any design strategy must cover. Analysis Period may be identical to the selected performance period. However, realistic practical performance limitations for some pavement designs may necessitate the

consideration of staged construction or planned rehabilitation to achieve the desired analysis period.

**4.3.3.3 Designed Life :** The designed life of the pavement is defined by the point when resurfacing is required at time intervals, shorter than a predetermined gap [3].



#### 4.3.4 Reliability :

Design reliability refers to the degree of certainty that a given design alternative will last for the entire analysis period. For a given reliability level, the reliability factor is a function of the overall standard deviation that accounts for standard variation in materials and construction, the probable variation in the traffic prediction, and the normal variation in pavement performance for a given design traffic application. The selection of standard deviation must be representative of local conditions. When staged construction is to be considered, it is important to recognize the need to compound the reliability of each individual stage of the strategy to achieve the desired overall reliability. The designed

reliability of each stage can be expressed as

$$R_{\text{stage}} = (R_{\text{overall}})^{1/n}$$

Where n = Number of stages

#### 4.3.5 Layer Coefficients ( $a_1$ ) :

The Layer Coefficients ( $a_1$ ,  $a_2$ , and  $a_3$  for surface, base and sub-base layers, respectively) express the relative ability of a material or a unit thickness of a given material, to function as a structural component of the pavement. Typical values of layer coefficients range between 0 to 0.6 for different layers. For example, 2 inches of a material with a layer coefficient of 0.20 is assumed to provide the same structural contribution as 1 inch of a material with a layer coefficient of 0.40.

Caution is recommended in selecting layer coefficients for asphalt concrete with modulus values exceeding 450,000 psi because their increase in stiffness is accompanied by increased susceptibility to thermal and fatigue cracking. For estimating layer coefficient,  $a_1$ , for Asphalt Concrete from its elastic (resilient) modulus,  $E_{AC}$ , following relationship may be used [4]:

$$a_1 = 0.4 * \log_{10} (E_{AC} / 435) + 0.44$$

where,  $E_{AC}$  = Elastic modulus for Asphalt Concrete at 68° F in KSI.

For estimating layer coefficient,  $a_2$ , for granular base material from its elastic (resilient) modulus,  $E_{base}$ , following relationship may used [4]:

$$a_2 = 0.249 (\log_{10} E_{base}) - 0.977$$

where,  $E_{base}$  = Elastic modulus for granular base material.

For estimating layer coefficient,  $a_3$ , for sub-base material from its elastic (resilient) modulus,  $E_{sub-base}$ , following relationship may used [4]:

$$a_3 = 0.227 (\log_{10} E_{sub-base}) - 0.839$$

where,  $E_{sub-base}$  = Elastic modulus for sub-base material.

#### 4.3.6 **Structural Number (SN) :**

It is an index number, which is subsequently converted to the thickness of various pavement layers. The Structural Number (SN) of a pavement section is defined as the sum of the products of the layer coefficients and thickness of each of the pavement layers,

$$SN = a_1 D_1 + a_2 D_2 + \dots$$

Where  $a_1, a_2, \dots$  = Layer Coefficients for surface, base ...etc.

layers, respectively.

$D_1, D_2, \dots$  = Depths/thickness for surface, base ...etc. layers,

respectively.

#### 4.3.7 **Drainage Coefficients( $m_i$ ) :**

The AASHTO procedure provides a means to adjust layer coefficients to take into account the effect of certain levels of drainage on pavement performance. The effect of drainage on all untreated layers of pavement below the surface is considered by multiplying the layer coefficients,  $a_i$ , by a modifying factor,  $m_i$ . This modifying factor,  $m_i$  is a function of the drainage characteristics of roadbed soil and the amount of time the soil remains in saturated condition.

The Structural Number equation modified for drainage becomes,

$$SN = a_1 D_1 + a_2 D_2 m_2 + a_3 D_3 m_3$$

Where  $a_i$  = Layer Coefficients of layers  $i$  ;

$D_i$  = Thickness of layers  $i$ ;

$m_i$  = Drainage modifying factor for layer  $i$ .

## COMPUTER PROGRAM

### ROADS AND ESTIMATING ITS COST/MATERIALS

#### 5.1 GENERAL

A computer program/system, named as "BASHROAD" is made in List Programming (LISP) Language for designing different layers of flexible pavement, using the AASHTO (American Association of State, Highway, and Transportation Officials) procedure for flexible pavement thickness design, and for calculation of material quantities required and the cost of materials and road section (complete code along with the comments, is attached as Appendix-C, page 84-123, describing the functions performed by each "Rule" and by each "Function"). The structure and the user friendly features have been designed to facilitate use of the program even by amateurs. This system consists of a production rule system (containing a structure to store the rules, a production rule memory, a production rule handler etc.), a set of production rules stored in the production rule memory, and a set of functions for obtaining input data. The production rules, which are stored in the production rule memory, are designed as per the structure defined in the production rule system.

#### 5.2 GENERAL WORKING OF "BASHROAD"

Production rules are defined to calculate different layer thicknesses of the road, material quantities required for construction of the road and cost of materials and the road



## CHAPTER 5

### **COMPUTER PROGRAM “BASHROAD” FOR DESIGNING**

### **ROADS AND ESTIMATING ITS COST/MATERIALS**

#### **5.1 GENERAL :**

A computer program/system, named as “BASHROAD” is made in List Programming (LISP) Language for designing different layers of flexible pavement, using the AASHTO (American Association of State, Highway, and Transportation Officials) procedure for flexible pavement thickness design, and for calculation of material quantities required and the cost of materials and road section (complete code along with the comments, is attached as Appendix-C, page 84-123, describing the functions performed by each “Rule” and by each “Function”). The structure and the user friendly features have been designed to facilitate use of the program even by amateurs. This system consists of a production rule system (containing a structure to store the rules, a production rule memory, a production rule handler etc.), a set of production rules stored in the production rule memory, and a set of functions for obtaining input data. The production rules, which are stored in the production rule memory, are designed as per the structure defined in the production rule system.

#### **5.2 GENERAL WORKING OF “BASHROAD” :**

Production rules are defined to calculate different layer thickness of the road, material quantities required for construction of the road and cost of materials and the road

section. In addition, all the production rules display important results on the screen including values of certain design parameters/variables, thickness of different layers of the road, material quantities for each layer and costs per lane-mile as well as for the entire road section. moved up or down with the help of scroll bar. In this way the user has the

access These production rules are stored in the production rule memory from where they can be accessed and used on requirement bases. A set of functions which is defined for obtaining input data from user, gets the data and checks if it is within specified limits. This check is to avoid any mistake during entering the data (i.e. typing mistake, entering characters along with a number etc.), or if user tries to enter an impossible value for any parameter. Range of possible values for any parameter is obtained from AASHTO (American Association of State, Highway, and Transportation Officials) Guide for Design of Pavement Structures, Washington, D.C., 1986 .

After checking its validity, the data is stored in the data base against their respective appropriate words. After completion of data, a function " RUN " is called to fire all the related/required rules in the production rule memory. The production rule handler provides interface between the production rule system/memory and other functions and applications and manages to fire all the related/required rules in the production rule memory. As a result of firing of these rules, all the s-expressions are executed i.e. in addition to calculating values of certain design parameters/variables, different layer thickness are calculated for the road, material quantities are calculated for each layer and costs are estimated per lane-mile as well as for the entire road section. Important results are then shown on the screen including values of certain design

parameters/variables, thickness of different layers of the road, material quantities for each layer and costs per lane-mile as well as for the entire road section.

This program gets input and prints the results in shape of a continuous screen which can be moved up or down with the help of scroll bar. In this way the user has the access to the input data as well as to the output results. Overall working of this program is elaborated with the help of two examples in the next chapter. Example 1 (Appendix-D, page 124-130 ) gives only the design results as the user was not interested in material quantities required for construction of the road and cost of materials/road where as example 2 (Appendix-E, page 131-139) shows all the results as the user desired so. These examples would be elaborated in the next chapter.

## CHAPTER 6

### **EXECUTION/COMPUTER ANALYSIS - BASHROAD**

#### **6.1 PROCEDURE TO START “ BASHROAD ” :**

This computer program can be run in the LISP environments on a Personal Computer. If working in UNIX environments, then it can be run by using emacs-lisp interface, which can be achieved with the help of a command “ **emacs -1.5.2** ” . LISP environments would be reached as shown in Appendix-D, page 124.

At the LISP prompt, the file “ *bashroad* ” , containing the source code can be loaded with the help of a command “ **(load “bashroad”)** ”, by making sure that this file is in the present working directory. Now the computer is ready to run this program. It can be started by giving command “ **(bashroad)** ” or “ **(start)** ” , as shown in Appendix-D, page 124.

#### **6.2 ENTERING INPUT DATA :**

As shown in Appendix-D and Appendix-E, after loading the file “ *bashroad* ” , this program is started by giving command “ **(bashroad)** ” at the LISP prompt. After giving some introductory remarks, following procedure is adapted in order to obtain the input data :-

- a. Enter required number of lanes.
- b. Enter required width of each lane.
- c. Enter required road length .



d. User is asked to select any of the following options.

- (1) To enter the structural number required to protect different layers.
- (2) The system should calculate SN values.
- (3) Exit from the program.

**Option 1** . (Option 1 is not shown in the examples) This option is given so that if the user wants to give SN values as input, which may have been calculated outside this program, he can do so. If the user selects this option, he is not asked to enter any data, required for calculation of SN values, i.e. the system would skip few steps ( i.e. steps e-p), and will ask the user about the asphalt concrete modulus of elasticity (as in step “ q ” below). In case of selecting this option, the user is asked to enter the following data ( instead of steps e-p):-

- (1). The structural number required to protect base layer.
- (2). The structural number required to protect subbase layer.
- (3). The structural number required to protect roadbed soil.

In this case the rule “ Start-from-a1 ” is fired which starts calculations from “ a1 ” instead of “ SN1 ”. From step “ q ” onwards, the procedure remains same for both options.

**Option 2** . (Option 2 is shown in example 1 and 2) This option is given so that if the user wants the system to calculate the values for SN1, SN2, and SN3, he can do so. If the user selects this option, he

is asked for data required for calculation of SN values. The procedure for option 2 is given below.

**Option 3** . (Option 3 is not shown in any of the two examples)

This option is given to exit from the system to LISP prompt.

As the user has given option 2 (i.e. The system should calculate SN values), the procedure would continue as from step “ e ” below.

e. Enter design period of the road in years.

f. Enter the equivalent single axle load (ESAL) ; For this purpose the user

is given two options:-

**Option 1** . ( Shown in example 2, Appendix-E ). This option is given so that the user can give the equivalent single axle load (ESAL) for design period of road. In this case the system would skip few steps ( i.e. steps g-h), as there is no need of getting data for calculating equivalent single axle load (ESAL) for design period, and will ask the user about the directional distribution factor for traffic (as in step “ i ” below). The procedure for option 2 is followed there after.

**Option 2** . ( Shown in example 1, Appendix-D ). This option is given so that the user can give the equivalent single axle load (ESAL) for the initial year only. In this case the system would continue as from step “ g ” below.

g. Enter the equivalent single axle load (ESAL) for the first year.

h. Enter annual growth rate of traffic.

i. Enter directional distribution factor for traffic.

- j. Enter lane distribution factor for traffic.
- k. Enter overall standard deviation.
- l. Enter design reliability in percentage.
- m. Enter overall serviceability loss OR initial and terminal pavement

serviceability index ; For this purpose for the user is given two options:-

**Option 1** . ( Shown in example 2, Appendix-E ). This option is given so that the user can give the overall serviceability loss. In this case the system would skip few steps( i.e. steps n-o), as there is no need of calculating overall serviceability loss, and will ask the user about the serviceability loss due to the environment. The procedure for option 2 is followed there after as from step “ p ” below..

**Option 2** . ( Shown in example 1, Appendix-D ). This option is given so that the user can give the initial and terminal pavement serviceability index. In this case the system would continue as from step “ n ” below.

- n. Enter the initial pavement serviceability index.
- o. Enter the terminal pavement serviceability index.
- p. Enter the serviceability loss due to the environment.
- q. Enter Asphalt concrete modulus of elasticity.
- r. Enter granular Base resilient modulus.
- s. Enter SubBase resilient modulus.
- t. Enter roadbed soil resilient modulus.

- u. Enter drainage modifying factor for Base layer.
- v. Enter drainage modifying factor for SubBase layer.
- w. Choose ‘ Y ’ or ‘ N ’ for material and cost estimation in addition to layer thickness.

**Option ‘ N ’**. ( Shown in example 1, Appendix-D ). This option is given so that the user can have only pavement design if he does not require the material and cost estimation. In this case the system would skip all the subsequent steps for data input, as they are only related with material and cost estimation, and would start giving the design results as shown in example 1, Appendix-D.

**Option ‘ Y ’**. ( Shown in example 2, Appendix-E ). This option is given so that the user can get the material and cost estimation in addition to pavement design. In this case the system would continue as from step “ x ” below.

- x. Enter the required compacted density of Asphalt Concrete layer.
- y. Enter the required compacted density of Base layer.
- z. Enter the required compacted density of SubBase layer.
- aa. Enter cost of the asphalt concrete mix (Dollars/Ton).
- bb. Enter cost of the base material (Dollars/Ton).
- cc. Enter cost of the subbase material (Dollars/Ton).
- dd. Enter the overhead-charges for one mile of each lane (As per the contract, if possible).



Now the system has obtained complete data and would start execution of the rules and displaying the results as shown in example 2, Appendix-E .

## CHAPTER 7

### **DESIGNING PROCEDURE USED IN PROGRAM**

#### **7.1 GENERAL :**

After obtaining complete data, the designing of road and material and cost estimation is done with the help of different functions/rules, defined for this purpose. Calculation procedure and the Equations used for some of the important design parameters/variables, thickness of different layers of the road, material quantities for each layer and costs per lane-mile as well as for the entire road section, are briefly explained in this chapter.

#### **7.2 Equivalent Single Axle Load (ESAL) :**

It represents the total number of equivalent 18-kip single axle load applications during the analysis period.

$$ESAL = [ ESAL_i * \{ (1+i)^n - 1 \} ] / i$$

Where ESAL = Total number of equivalent 18-kip single  
axle load applications during analysis period.

n = Analysis period (years)

ESAL<sub>i</sub> = Number of equivalent 18-kip single axle  
load applications during initial year.

i = Annual growth rate of traffic.

### 7.3 **Load Condition( $W_{18}$ ) :**

It is a proxy of prevailing loading conditions for the pavement to be designed and it represents the total number of design equivalent 18-kip single axle load applications during the analysis period.

$$W_{18} = \text{ESAL} * \text{DD} * \text{LD}$$

Where  $W_{18}$  = Design Load represented in equivalent  
18-kip single axle load applications.

DD = Directional distribution factor for traffic.

LD = Lane distribution factor for traffic.

### 7.4 **Overall Serviceability Loss ( $\Delta\text{PSI}_o$ ) :**

It is the difference between initial pavement serviceability index and terminal pavement serviceability index.

$$\Delta\text{PSI}_o = \text{PSI}_i - \text{PSI}_t$$

Where  $\Delta\text{PSI}_i$  = Initial pavement serviceability index.

$\text{PSI}_t$  = Terminal pavement serviceability index.

### 7.5 **Design Serviceability Loss ( $\Delta\text{PSI}$ ) :**

It is the design serviceability loss due to traffic, and is equal to the difference between values of overall serviceability loss and serviceability loss due to the environment.

$$\Delta\text{PSI} = \text{PSI}_o - \text{PSI}_{\text{env}}$$

Where  $\Delta\text{PSI}_o$  = Overall serviceability loss.

$\Delta\text{PSI}_{\text{env}}$  = Serviceability loss due to the environment.

## 7.6 Required Structural Number (SN) :

The required Structural Numbers, SN1, SN2 and SN3 to protect the base layer, subbase layer and the roadbed soil respectively are calculated with the help of following equation :-

$$\log_{10}(W_{18}) = Z_R * S_O + 9.36 * \log_{10}(\text{SN}+1) - 0.20 + \frac{\log_{10}[\Delta\text{PSI} / (4.2 - 1.5)]}{0.4 + [1094 / (\text{SN} + 1)^{5.19}]} + 2.32 * \log_{10} M_R - 8.07$$

Where  $W_{18}$  = Design Load represented in equivalent 18-kip single axle load applications.

$Z_R$  = Standard normal deviate corresponding to the design reliability (obtained by using a table, stored in the program).

$S_O$  = Overall Standard Deviation.

SN = Required Structural Number.



$\Delta PSI$  = Design serviceability loss.

$M_R$  = Effective Resilience Modulus of the layer to be protected.

### 7.7 **Layer Coefficient for AC ( $a_1$ ) :**

It is calculated by using the following equation :-

$$a_1 = 0.4 * \log_{10} ( E_{AC} / 435 ) + 0.44$$

where,  $E_{AC}$  = Elastic modulus for Asphalt  
Concrete at 68° F in KSI.

### 7.8 **Layer Coefficient for granular base material ( $a_2$ ) :**

The layer coefficient for granular base material is calculated with the help of following equation :-

$$a_2 = 0.249 (\log_{10} E_{base}) - 0.977$$

where,  $E_{base}$  = Elastic modulus for granular base material.

### 7.9 **Layer Coefficient for subbase material ( $a_3$ ) :**

The layer coefficient for subbase material is calculated with the help of following equation :-

$$a_3 = 0.227 (\log_{10} E_{\text{sub-base}}) - 0.839$$

where,  $E_{\text{sub-base}}$  = Elastic modulus for subbase material.

### 7.10 **Thickness of the AC layer ( $D_1^*$ ) :**

For finding thickness of the AC layer, first calculated thickness is obtained and then it is rounded up to find the designed thickness. Rounding up/down is done because it may not be possible in the field to achieve precision up to fraction of an inch, during road construction. Calculated thickness is obtained with the help of following equation :-

$$D_1 = SN_1 / a_1$$

Where  $D_1$  = Calculated thickness of the AC layer.

$SN_1$  = Required Structural Number to  
protect the base layer.

$a_1$  = Layer coefficient for AC.

The designed thickness of AC layer ( $D_1^*$ ) is obtained by :

$$D_1^* = \text{Rounded up } D_1$$

**Check**

If  $SN_1^* \geq SN_1$  Then  $D_1^*$  is OK, else increase  $D_1^*$  by one inch and apply the same check again until this condition is satisfied.

$$\text{Here } SN_1^* = (a_1) * (D_1^*)$$

OR = Actual Structural Number obtained  
by providing AC thickness equal to  $D_1^*$  .

**7.11 Thickness of the base layer ( $D_2^*$ ) :**

Calculated thickness of the base layer is obtained with the help of following equation :-

$$D_2 = (SN_2 - SN_1^*) / (a_2 * m_2)$$

Where  $D_2$  = Calculated thickness of the base layer.

$SN_2$  = Required Structural Number to protect the  
subbase layer.

$a_2$  = Layer coefficient for base material.

$SN_1^*$  = Actual Structural Number obtained by  
providing AC thickness equal to  $D_1^*$  .

$m_2$  = Drainage modifying factor for base layer.

The designed thickness of base layer ( $D_2^*$ ) is obtained by rounding up/down and applying the check in order to ensure that the required Structural Number to protect the subbase layer is achieved i.e.

$$D_2^* = \text{Rounded up/down } D_2$$

**Check**

If  $(SN_1^* + SN_2^*) \geq SN_2$  Then  $D_2^*$  is OK, else increase  $D_2^*$  by one inch and apply the same check again until this condition is satisfied.

$$\text{Here } SN_2^* = a_2 * m_2 * (D_2^*)$$

OR  $SN_2^* =$  Actual Structural Number obtained by providing base thickness equal to  $D_2^*$ .

**7.12 Thickness of the subbase layer ( $D_3^*$ ) :**

Calculated thickness of the subbase layer is obtained with the help of following equation :-

$$D_3 = (SN_3 - SN_1^* - SN_2^*) / (a_3 * m_3)$$

Where  $D_3 =$  Calculated thickness of the subbase layer.

$SN_3 =$  Required Structural Number to protect the roadbed soil.

$a_3 =$  Layer coefficient for subbase material.

$SN_1^* =$  Actual Structural Number obtained by providing AC thickness equal to  $D_1^*$ .

$SN_2^* =$  Actual Structural Number obtained by providing base thickness equal to  $D_2^*$ .

$m_3 =$  Drainage modifying factor for subbase layer.



The designed thickness of subbase layer ( $D_3^*$ ) is obtained by rounding up/down and applying the check in order to ensure that the required Structural Number to protect the roadbed soil is achieved i.e.

$$D_3^* = \text{Rounded up/down } D_3$$

### Check

If  $(SN_1^* + SN_2^* + SN_3^*) \geq SN_3$  Then  $D_3^*$  is OK, else increase  $D_3^*$  by one inch and apply the same check again until this condition is satisfied.

$$\text{Here } SN_3^* = a_3 * m_3 * (D_3^*)$$

OR  $SN_3^* = \text{Actual Structural Number obtained by providing subbase thickness equal to } D_3^* .$

### **7.13 Compacted Volume of Asphalt Concrete Mix :**

The total compacted volume (in  $\text{ft}^3$ ) of Asphalt Concrete mix is calculated with the help of following equation :-

$$V_{AC} = N * W * L * 5280 * (D_1^* / 12)$$

Where  $V_{AC}$  = Total compacted volume of  
Asphalt Concrete Mix in  $\text{ft}^3$

$N$  = Number of lanes.

$W$  = Width of each lane in feet.

$L$  = Length of the road in miles.

$D_1^*$  = Design thickness of the AC  
layer in inches.

#### 7.14 **Compacted Volume of Base material :**

The total compacted volume (in  $\text{ft}^3$ ) of base material is calculated with the help of following equation :-

$$V_{\text{base}} = N * W * L * 5280 * (D_2^* / 12)$$

Where  $V_{\text{base}}$  = Total compacted volume of  
base material, in  $\text{ft}^3$

$N$  = Number of lanes.

$W$  = Width of each lane in feet.

$L$  = Length of the road in miles.

$D_2^*$  = Design thickness of the base  
layer in inches.

#### 7.15 **Compacted Volume of SubBase material :**

The total compacted volume (in  $\text{ft}^3$ ) of subbase material is calculated with the help of following equation :-

$$V_{\text{s.base}} = N * W * L * 5280 * (D_3^* / 12)$$

Where  $V_{s,base}$  = Total compacted volume of  
subbase material, in  $ft^3$

$N$  = Number of lanes.

$W$  = Width of each lane in feet.

$L$  = Length of the road in miles.

$D_3^*$  = Design thickness of the  
subbase layer in inches.

#### **7.16 Weight of Asphalt Concrete Mix :**

The total weight (in tons ) of asphalt concrete mix is calculated with the help of following equation :-

$$W_{AC} = (V_{AC} * D_{AC}) / 2240$$

Where  $W_{AC}$  = Total weight of asphalt  
concrete mix in tons.

$V_{AC}$  = Total compacted volume of  
asphalt concrete mix in  $ft^3$ .

$D_{AC}$  = Compacted density of  
asphalt concrete layer in pcf.

**7.17 Weight of Base material :**

The total weight (in tons ) of base material is calculated with the help of following equation :-

$$W_{\text{base}} = (V_{\text{base}} * D_{\text{base}}) / 2240$$

Where  $W_{\text{base}}$  = Total weight of base material in tons.

$V_{\text{base}}$  = Total compacted volume of base material in ft<sup>3</sup>.

$D_{\text{base}}$  = Compacted density of base layer in pcf.

**7.18 Weight of SubBase material :**

The total weight (in tons ) of subbase material is calculated with the help of following equation :-

$$W_{\text{s.base}} = (V_{\text{s.base}} * D_{\text{s.base}}) / 2240$$

Where  $W_{\text{s.base}}$  = Total weight of subbase material in tons.

$V_{\text{s.base}}$  = Total compacted volume of subbase material in ft<sup>3</sup>.

$D_{\text{s.base}}$  = Compacted density of subbase layer in pcf.



**7.19 Cost of Asphalt Concrete Mix :**

The total cost (in dollars ) of asphalt concrete mix is calculated with the help of following equation :-

$$C_{AC} = (W_{AC} * P_{AC})$$

Where  $C_{AC}$  = Total cost of asphalt  
concrete mix in dollars.

$W_{AC}$  = Total weight of asphalt  
concrete mix in tons.

$P_{AC}$  = Market price of asphalt  
concrete mix in dollars/tons.

**7.20 Cost of Base material :**

The total cost (in dollars ) of base material is calculated with the help of following equation :-

$$C_{base} = (W_{base} * P_{base})$$

Where  $C_{base}$  = Total cost of base material  
in dollars.

$W_{base}$  = Total weight of base  
material in tons.

$P_{base}$  = Market price of base  
material in dollars/tons.

### 7.21 **Cost of SubBase material :**

The total cost (in dollars) of subbase material is calculated with the help of following equation :-

$$C_{s.base} = (W_{s.base} * P_{s.base})$$

Where  $C_{s.base}$  = Total cost of subbase material in dollars.

$W_{s.base}$  = Total weight of subbase material in tons.

$P_{s.base}$  = Market price of subbase material in dollars/tons.

### 7.22 **Cost of each lane mile :**

The cost (in dollars) per lane mile of road is calculated with the help of following equation :-

$$C_{LM} = \{ (C_{AC} + C_{base} + C_{s.base}) / (N * L) \} + C_{OHLM}$$

Where  $C_{LM}$  = Total cost of one mile of each lane in dollars.

$C_{base}$  = Total cost of base material in dollars.

$C_{s.base}$  = Total cost of subbase material in dollars.

$N$  = Number of lanes.

$L$  = Length of the road in miles.

$C_{\text{OHLM}}$  = Overhead cost/charges for  
one mile of each lane in  
dollars.

### 7.23 **Total cost of the road :**

The total cost (in dollars ) of designed road section is calculated with the help of following equation :-

$$C_T = C_{LM} * N * L$$

Where  $C_T$  = Total cost of designed road  
section in dollars.

$C_{LM}$  = Total cost of one mile of  
each lane in dollars.

$N$  = Number of lanes.

$L$  = Length of the road in miles.

## **CHAPTER 8**

### **ACCOMPLISHMENTS AND RECOMMENDATIONS**

#### **8.1 ACCOMPLISHMENTS :**

Several accomplishments have been achieved successfully in this study, in the field of road designing and cost analysis, by making a computer program named as BASHROAD , in an Artificial Intelligence language i.e. LISP. These include :

1. Road designing is introduced in the field of Artificial Intelligence and programming is done in List Programming(LISP) Language, which may prove to be a door opening step for the future research. Previously the programming in the field of road designing and cost analysis was being done in other computer languages, like C, C++, etc. which are not versatile as that of Artificial Intelligence languages.
2. A sound foundation is provided for future programming by designing a production rule system. This production rule system makes the programming easy and provides to the programmer, the flexibility of programming and ability to expand/change at any stage.
3. A structure to define the rules is designed and a number of production rules has been defined which calculate and display important results on the screen including values of certain design parameters/variables, different layer thickness of the road, material quantities required for its construction, material cost and the cost per lane mile and for entire road section. Any number of production rules defined, for

further programming, by following the same structure would perform the functions as per programmer's requirement.

4. A production rule memory is designed which can store any number of production rules. The rules can be added to or deleted from this memory at any time, providing the flexibility of programming to the programmer. These rules can be accessed at any time during execution of the program as per programmer's requirement.
5. A production rule handler is designed which provides the interface between the " functions " and the production rule system and the production rule memory. With the help of this production rule handler, the programmer can access and execute any production rule at any time during execution of the program.
6. This computer program, BASHROAD can design pavement thickness successfully and accurately, using AASHTO procedure for flexible pavement thickness design.
7. BASHROAD also gives SN values, layer coefficient values and some other design parameters, material quantities required for construction of the designed road, material cost and the cost per lane mile and for entire road section.
8. The computer program BASHROAD also checks if the data entered by user is within specified limits. This check is to avoid any mistake during entering the data or if user tries to enter an impossible value for any parameter. Range of possible values for any parameter is obtained from AASHTO Guide for Design of Pavement Structures, 1986 .



## **8.2    RECOMMENDATIONS FOR FUTURE RESEARCH :**

Based on the results and accomplishments of this research study, the following recommendations are made :-

1.    Environmental impact, seasonal variations, frost heave potential and some other minute considerations should be incorporated in the program to make it more useful and accurate.
2.    Cost considerations in selection of layer thickness should also be incorporated in the program so that the initial cost of the pavement can be minimized by providing one alternative design for each combination of materials being considered. In this process, thickness of one layer can be increased to reduce the thickness of another layer keeping in view their cost effectiveness.
3.    This system should be connected to another application, preferably a GIS application such that it should perform functions in the following manner :-
  - a.    When a user marks two points on the digital map, apply Artificial Intelligence search techniques in order to find different suitable alignments by taking into account any barriers, constraints, slopes, buildup/urban areas, rivers, streams, etc.
  - b.    Obtain part of the data from the user (as being obtained currently), and part of the data from the digital map which may include length of road required to be constructed for different alignments, ground slopes, under ground water table, barriers, road bed soil resilience modulus, soil types etc. For this purpose some additional coverage's (i.e. road bed soil resilience modulus-coverage etc.) may have to be digitized because they are not

being digitized in usual practice. Environmental impact and seasonal variations should also be considered.

- c. Complete data should be provided to this program in order to get the design results and cost estimations for each alignment.
- d. At the end, the best alignment should be found by comparing the lengths, cost, materials and the effort required for all the alignments.

**Appendix-A**

```

;
;
; PRODUCTION RULES SYSTEM
;-----;
;          STRUCTURE TO STORE THE RULE          ;
;          ;                                     ;
;          ;                                     ;
; The structure used to store the rule is a list of rule-name, pattern, ;
; list of variables in pattern and the list of S expressions :-      ;
;          ;                                     ;
; ( rule-name pattern (!var1 !var2 ...!varN) (S-exp1 S-exp2 ... S-expN) ) ;
;          ;                                     ;
; STORERULE is called to store this structure as a global value in the ;
; rule-name .                                     ;
;-----;
;;;
;;;          RULE
;;;
;;;
;;; Arguments : priority      ; number between 0 - 400 (inclusive)
;;;           : rule-name    ; symbol to be assigned structure
;;;           : pattern      ; logical term(rule fires if it matches)
;;;           : arrow        ; seperator in rule ( '-->' )
;;;           : s-exp        ; list of s-expressions to be executed
;;;           ; when rule fires
;;;
;;;
;;; Result      : same as the result returned from storerule (function)
;;; Sideeffects : calls storerule which assigns the structure of rule to
;;;           : rule-name as its global value.
;;;
;;;
;;; It checks if the syntax of the rule is correct by calling syntax-bad, if
;;; the syntax is not correct it returns with error message.
;;; If the syntax is correct it calls storerule which assigns the stucture to
;;; rulename.
;;; function CREATE-VARS is called to get the list of variables used
;;; in pattern, as it is useful and more speed efficient for varaible bindings
;;; when the rule is being used in TRYRULE.

(defmacro rule (priority rule-name pattern arrow &rest s-exps)
  `(if (syntax-bad ',priority ',rule-name ',pattern ',arrow)
      (error "SYNTAX ERROR IN THE RULE MACRO ****")
      (storerule ',rule-name ',priority
                  (list ',rule-name ',pattern
                        ',(create-vars pattern nil) ',s-exps)))
  )
)

```

```

;;;
;;;
;;;          CREATE-VARS
;;;
;;; Arguments : pattern      ; whose variables are to be collected
;;;           : var-list     ; list of variables (initially nil)
;;; Result    : list of variables
;;;
;;; It searches the pattern list and conses all variables found to var-list.

```

```

(defun create-vars (pattern var-list)
  (cond ((is-var pattern) (cons pattern var-list))
        ((atom pattern) var-list)
        (t (create-vars (cdr pattern)
                         (create-vars (car pattern) var-list))))
  )
)

```

```

;;;-----;
;;;
;;;
;;;          TRYRULE
;;;
;;;
;;; Arguments : rule-name      ;rule to be tried.
;;; Result    : t / nil       ;if rule fires / does not fire
;;;
;;; It initially calls DB-FIND to get the bindings list of all matches in data
;;; base. if binding list is empty it returns nil. else ..
;;; It cdrs down the bindings list for each match's individual binding and
;;; creates a LET structure with Backquote to locally bind the variables to
;;; their bindings. Function CREATE-BINDINGS is used to create the variable
;;; assignments list for LET structure. Once the structure is complete it is
;;; EVALuated. ,@ is used to remove the outer brackets of s-exps list.
;;;

```

```

(defun tryrule (rule-name)
  (let ((bind-list (db-find (cadr rule-name)))
        (variables (caddr rule-name))
        (s-exps (caddr rule-name)))
    (if (not bind-list) nil ;no match

```

```

      (dolist (binding bind-list t)
        (eval `(let ,(create-bindings variables (car binding))
                  ,@s-exps))))
    )
)

```

```

;;;
;;;          CREATE-BINDINGS
;;;
;;; Arguments  : vars          ;list of variables from rule structure
;;;             : table         ;table of bindings of one match
;;; Result     : list of variable assignments for LET structure
;;;             : e.g ((!x a) (!y b) (!z c))
;;;
;;; It recurses down the list of variables and for each variable finds the
;;; binding from table using FINDENTRY function and quotes it using backquote
;;; trick to get the assignment work for the binding not bindings value since
;;; the function LIST will try to evaluate its arguments.

```

```

(defun create-bindings (vars table)
  (if (null vars) nil
      (cons (list (car vars) `(quote ,(findentry (car vars) table)))
            (create-bindings (cdr vars) table))
  )
)

```



```

-----;
;           A PRODUCTION RULE HANDLER           ;
;                                           ;
; This production rule handler is based on the function and macro :      ;
; TRYRULE & RULE defined earlier, and with them as basic tools to define the ;
; production rules and then to try them,                                ;
; It defines a function RUN which allows the rules (stored in a production ;
; memory called RULES) to be tried one after another, based on their priority;
; and recency.                                                           ;
-----;

```

```

;;;           STORERULE
;;;
;;; Arguments : name                (name of the rule defined)
;;;           : priority            (0-400 rule priority)
;;;           : structure           (to be assigned to rule name)
;;; Result    : P-list under property of priority
;;; SideEffects : assigns structure to rule name as global value, and inserts
;;;           : the rule name in one of the property lists of RULES under
;;;           : property "priority" as passed to RULE macro.
;;;
;;; It stores the rules in a production memory (called RULES) and assigns
;;; the structure of rule to the rule-name. The rule is inserted at the end of
;;; p-list so that the oldest rule can be tried first.

```

```

(defun storerule (name priority structure)
  (set name structure)
  (setf (get 'rules priority) (append (get 'rules priority)(list name)))
)

```

```

;;;           HALT
;;;
;;; Arguments : result                (result which is to be passed)
;;;           :                       (back. It is optional      )
;;; Result    : If argument is passed then that evaluated argument
;;;           : Else nil
;;;
;;; It throws back the optional argument else nil to the CATCH in RUN function

```

```
(defun halt (&optional (result nil))
  (throw 'result-val result)
)
```

```
;;;          RUN
;;;
;;; Arguments  : none
;;; Results    : nil / passed by HALT
;;; SideEffects : As resulting by the actions in firing rules.
;;;
;;; It tries all rules in Production memory, starting from highest priority
;;; rules (oldest of them). if any rule fires, it starts from the beginning,
;;; else it exits with nil. if HALT is executed in any firing rule it exits
;;; with the result of HALT if any.
```

```
(defun run ()
  (catch 'result-val
    (do ((priority 400 (1- priority)))
      ((minusp priority) nil)
      (if (rule-fired (get 'rules priority)) (setq priority 401)))
    )
  )
)
```

```
;;;          RULE-FIRED
;;;
;;; Arguments  : list of rules          (one p-list at same priority)
;;; Result     : T if any rule fires    (else nil)
;;; SideEffect : none
;;;
;;; It cdr's down the p-list and tries every rule as it goes down, if the rule
;;; fires it returns T without trying any further rules.
```

```
(defun rule-fired (rule-list)
  (cond ((null rule-list) nil)
        ((tryrule (eval (car rule-list))) t)
        (t (rule-fired (cdr rule-list))))
  )
)
```

```

;;;-----;
;;;
;;;
;;;          SYNTAX-BAD
;;;
;;;
;;; function which checks if the syntax of rule macro is wrong or not

```

```

(defun syntax-bad (priority rule-name pattern arrow)
  (cond ((not (numberp priority)) (princ "priority is not number" )
        ((or (< priority 0)
              (> priority 400)) (princ "priority not in limits" )
        ((not (symbolp rule-name)) (princ "rule-name not symbol" )
        ((not (well-formed pattern)) (princ "pattern not logical" )
        ((not (equal arrow '-->)) (princ "arrow not correct" )
  )
)

```

```

;;;
;;;
;;;          IS-VAR
;;;
;;;
;;; checks if the term is a variable with '!' sign infront.

```

```

(defun is-var (term)
  (and (symbolp term) (has-e-mark term))
)

```

```

;;;
;;;
;;;          WELL-FORMED
;;;
;;;
;;; functions which check if the term is well formed or not.
;;; If its a list and its car is function and rest of term if formed its ok

```

```

(defun well-formed (term)
  (and (listp term) (function-ok (car term)) (rest-formed term))
)

```

```

;;;           Function-ok
;;;
;;; checks if the function is a variable or number or a list
;;; then it returns nil else true.

```

```

(defun function-ok (predicate)
  (not (or (listp predicate) (numberp predicate)
           (and (symbolp predicate) (has-e-mark predicate)))
  )
)

```

```

;;;           Rest-formed
;;;
;;; checks if the term after the function is ok.

```

```

(defun rest-formed (term)
  (if (atom term) t
      (and (if (listp (car term))
                (and (function-ok (caar term))
                     (rest-formed (car term)))
                t)
           (rest-formed (cdr term)))
  )
)

```

```

;;;           Term Unification
;;;
;;; A logical term is represented by a list whose first element
;;; is the functor-name (represented as an alphanumeric Lisp symbol)
;;; and whose other elements (if any) are the arguments of the term,
;;; in order.
;;; Unification variables are represented as symbols whose first
;;; character is an exclamation-mark; the predicate "has-e-mark"
;;; is used to detect such symbols.

```

```

;;; functions for tables should also be loaded from file tables.lsp
;;; function has-e-mark should be loaded from file emark.lsp

```

```

;;; Functions used from elsewhere:
;;;
;;; has-e-mark - returns t if its argument is a symbol that

```

```

;;;          starts with an exclamation-mark, nil otherwise
;;;          (error if not a symbol)
;;;  newtable    - constructs a new, empty, table
;;;  addentry    - adds an entry to a table
;;;  hasentry    - tests if an item appears in a table
;;;  findentry   - extracts the value entered against a key

```

```

;;;
;;;          UNIFY
;;;
;;; takes two logical terms and checks if they are
;;; well formed, & function is a symbol. if not prints
;;; error. else it passes the terms to function MATCH
;;; with a list of newtable for the bindings of the
;;; variables present.
;;; returns nil if match fails
;;; or list of table if succeeds.

```

```

(defun unify ( term1 term2 )
  (cond (( not (and (if (listp term1) (function-ok(car term1)) t)
                    (well-formed term1)))
        (error " Term 1 is not well formed **"))

        (( not (and (if (listp term2) (function-ok(car term2)) t)
                    (well-formed term2)))
        (error " Term 2 is not well formed **"))

        (t ( match term1 term2 (list (newtable))))
  )
)

```

```

;;;          MATCH
;;;
;;; checks if term1 is a variable, then matches the variable
;;; by calling MATCH-VAR. if term2 is a variable, then calls
;;; MATCH-VAR with term2 as var. else checks if terms are atoms
;;; if nor atoms nor variables then processes the car's and cdr's
;;; of both terms and extending the table if variables found in
;;; any part.

```

```

(defun match ( term1 term2 table )
  ( cond ((is-var term1) (match-var term1 term2 table))
        ((is-var term2) (match-var term2 term1 table))
        ((atom term1) (if (eq term1 term2) table ))
        ((atom term2) nil)
        ( t (and (setq car-table (match (car term1) (car term2) table))
                  (match (cdr term1) (cdr term2) car-table)))
  )
)

```

```

;;;          MATCH-VAR
;;;
;;; checks if variable is same as second term, then only
;;; returns table. else it calls FINDENTRY to check if
;;; variable is already bound to some other variable. if
;;; it is bound then it calls MATCH again to match the bound
;;; variable with the other term. if not bound then it checks
;;; if the variable is contained in the term being matched to
;;; by calling CONTAINS. if it is not then it adds the bound
;;; pair in the table.

```

```

(defun match-var (var term table)
  ( if (equal var term) table
    (let ((bound-to (findentry var (car table))))
      (cond (bound-to (match bound-to term table))
            ((not (contains var term table))
             (list (addentry var term (car table)))))
    )
  )
)

```

```

;;;          CONTAINS
;;;
;;; if the term is itself a variable, then checks if same as
;;; variable , or checks if the term is bound then does that
;;; bound term or variable contain the parameter var, this
;;; is done recursively. else if the term is a list then
;;; recurses in the car's and cdr's of the term to check the
;;; circularities.

```



```

(defun contains (var term table)
  (cond ((is-var term) (or (equal var term)
                           (contains var (findentry term (car table))
                                       table)))
        ((atom term) nil)
        (t (or (contains var (car term) table)
                (contains var (cdr term) table))))
  )
)

```

```

;;; HAS-E-MARK

```

```

(defun has-e-mark (check-var)
  (eq #\! (char (string check-var) 0))
)

```

```

;-----;
;          STRUCTURE OF PROPERTY LISTS          ;
;          ;                                     ;
;          ;                                     ;
; The Data Base is based on the property lists of the primary & secondary ;
; keys of the individual terms inserted. The property lists have the ;
; following structure :- ;
;   plist --> ( No of terms (term1) (term2) (term3) ... (term n)) ;
;          ;                                     ;
;   position  plist ;
; Key --> _____ ;
;   | 1 | ---> | ( n t1 t2 ...tn) ;
;   ----- ;
;   | 2 | ---> | ( m t1 t2 ...tm) ;
;   . . . ;
;   . . . ;
;   . . . ;
;   . . . ;
;-----;

```

```

;;;
;;;
;;;
;;;
;;; Arguments : logical term      (to be inserted in indexed Data Base)
;;; Result    : Inserted term
;;; Side Effects: Pointers to term are added to individual position plists of
;;;              : keys & secondary keys (where position corresponds to the
;;;              : occurring position of the relative key ).
;;;              : Also the plist length counter is incremented.
;;;
;;;
;;; It cdr's down the term and calls insert to insert the pointer in each
;;; symbols indicated position.

```

```

(defun DB-INSERT (term)
  (cond ((not (well-formed term))
    (error " Term being INSERTED is not well formed ***")))

  (t (do ((position 1 (1+ position))      ; do local
          (key-term term (cdr key-term)))  ; variables
    ((null key-term) term)                 ; exit term
    (insert term (car key-term) position)  ; insert in
    ))                                     ; data base
  )
)

```

```

;;;
;;;
;;;
;;; Insert
;;;
;;; It calls Add-plist after distinguishing whether the key passed
;;; is a symbol or a list.

```

```

(defun insert (term key position)
  (cond ((symbolp key) (add-plist term key position)) ;if symbol ?
        ((listp key) (add-plist term (car key) position)) ;if list ?
  )
)

```

```

;;;
;;;
;;;
;;; Add-Plist
;;;
;;; Arguments : term      (to be inserted)
;;;            : key      (in whose plist the term is to be inserted)
;;;            : position (where the key occurred in the term)
;;; Side Effects: It actually updates the plist.

```

```

(defun add-plist (term key position)
  (setf (get key position)          ;get plist
        (cons (1+ (if (null (get key position)) 0      ;increment
                      (car (get key position))))      ;list length
              (cons term (cdr (get key position)))))    ;add pointer
  )
)

```

-----;

```

;;;          DB-DELETE
;;;
;;;
;;; Arguments  : logical term      ( used to delete any term which matches)
;;; Results    : list of all deleted terms
;;; Side Effects: any term which matches given term is removed from the plist
;;;              : of the primary key , and all pointers to it from plists of
;;;              : secondary keys are also deleted.
;;;              : from every plist if a term is removed its length is also
;;;              : decremented.
;;;
;;; It cdr's down the plist of primary key of the given term, whenever a match
;;; occurs (i.e unify returns non nil list) that term is removed plus all
;;; secondary indexing pointers are also removed.

```

```

(defun DB-DELETE (term)
  (cond ((not (well-formed term))
        (error "Term being DELETED is not well formed ***"))
        ((null (get (car term) 1)) nil)
        (t (do ((plist (cdr (get (car term) 1)) (cdr plist))
                (result-list nil))
                ((null plist) result-list) ;;; list of all deleted terms

            (cond ((unify term (car plist))
                   (setq result-list (cons (car plist) result-list))
                   (delete-secondary-indexes (car plist)
                                              (cdr (car plist)))
                   (remove-term (car plist) (car term) 1))
                  )
          )
        ))
)

```

```
;;; Remove-term
```

```
;;;
```

```
;;; Side Effects: the term is removed from the plist of symbol at index
```

```
;;; : 'position' and the length of plist is decremented.
```

```
(defun remove-term ( term symbol position )
  (setf (get symbol position) ;get plist
        (cons (1- (car (get symbol position))) ;decrement
              (remove term (cdr (get symbol position))) ;remove trm
              )
        )
  )
```

```
;;; Delete-secondary-indexes
```

```
;;;
```

```
;;; using the term being deleted by DB-DELETE it starts deleting it from
```

```
;;; the plists of its secondary keys. (i.e using cdr of term as key-term)
```

```
;;; it cdr's down key-term and removes term from plists (at positon index).
```

```
(defun delete-secondary-indexes ( term key-term)
  (do ((position 2 (1+ position)) (keys key-term (cdr keys)))
      ((null keys)
       (cond ((symbolp (car keys)) (remove-term term (car keys) position))
             ((listp (car keys)) (remove-term term (caar keys) position)))
      )
  )
```

```
-----;
```

```
;;; DB-FIND
```

```
;;;
```

```
;;; Arguments : logical term (used to find any term which matches it).
```

```
;;; Results : list of all matching items' binding lists.
```

```
;;; Side Effects: none.
```

```
;;;
```

```
;;; It finds the shortest plist of the keys' plists, and start matching the
```

```
;;; given term with the terms in the plist. Whenever a term matches (i.e unify
```

```
;;; returns non nill list) it cons's the binding list to the result list.
```

```

(defun DB-FIND (term)
  (cond ((not (well-formed term))
    (error "Term being FOUND is not well formed ***"))
    (t (do ((plist (find-shortest-plist term) (cdr plist)) ;do local
      (result-list nil) (bindings nil)) ;variables
      ((null plist) result-list) ;exit term
      (setq bindings (unify term (car plist))) ;if match
      (if bindings (setq result-list ;add to
        (cons bindings result-list))) ;result
      ))
    )
  )

;;;          find-shortest-plist
;;;
;;;
;;; Argument   : term
;;; Result     : shortest plist of keys in term ( with length removed )
;;;
;;; It cdr's down the term and for each key's plist and the previous shortest
;;; plist gets the shorter from function compare-length.

(defun find-shortest-plist (term)
  (do ((plist (get (car term) 1)) (position 2 (1+ position))
    (key-term (cdr term) (cdr key-term)))
    ((null key-term) (cdr plist))
    (setq plist (compare-length plist (car key-term) position))
  )
)

;;;          Compare-length
;;;
;;;
;;; Argument   : plist      (previous shortest plist)
;;;             : symbol,position (whose plist at 'position' is to be compared)
;;; Result     : shorter of the two plists
;;;
;;; After checking if the symbol is not a variable (with function-ok) or list
;;; and it has a plist for particular position, checks which plist is shorter.
;;; It does the same if symbol is a list, then it checks for is car.
;;; if symbol is a variable then it returns previous shortest plist.

```

```

(defun compare-length (plist symbol position)
  (cond ((and (function-ok symbol) (numberp (car (get symbol position)))
    (> (numberize (car plist)) (numberize(car (get symbol position)))))
    (get symbol position))

    ((and (listp symbol) (numberp (car (get (car symbol) position)))
    (> (numberize (car plist))
      (numberize (car (get (car symbol) position)))))
    (get (car symbol) position))

    (t plist)
  )
)

```

```

;;;          Numberize

```

```

(defun numberize (x)
  (if (numberp x)
      x
      0))

```



```

;
;           Table data structure           ;
; The table data structure is designed in the form of lists of individual ;
; entries in the list of table i.e           ;
; Table = ( (key1 value1) (key2 value2) ..... (keyN valueN) )           ;
; entry can be found by taking car of car of table           ;
; value can be found by car of cdr of car of table           ;
;
;

```

```
... NEWTABLE
```

[illegible]

ADDENTRY

(defun ADDENTRY ( K V D )	; function AddEntry
(cons	; creates a new table and
(list K V) D	; inserts K & V as a list plus
)	; the table D by using cons
)	

Findentry

(defun FINDENTRY (K D)	; function FindEntry
(cond	; check if table is
((EMPTY D) nil)	; Empty or End then return nil
((equal (caar D) K) (cadar D))	; Has key then return value
(t (FINDENTRY K (cdr D))))	; Else recurse till key found
)	; or End of table reached
)	


**HASENTRY**

(defun HASENTRY (K D)	; function HasEntry
(cond	; check if table is
((EMPTY D) nil)	; Empty or End then return nil
((equal (caar D) K) t)	; Has key then return true
(t (HASENTRY K (cdr D))))	; Else recurse till key found
)	; or End of table reached
)	

;;;                   EMPTY

(defun EMPTY (D)  
  (null D)  
)

; function Empty  
; If list null then return T

**Appendix-B****EXAMPLE****FUNCTION OF PRODUCTION RULE SYSTEM**

It is assumed that values of the variables, required for calculation purpose are obtained from the data base.

**DEFINING RULES :**

;INSERT A MINI-DATABASE

```
(mapcar #'db-insert '(
(road overhead charges)
(lanes -- --)
))
```

**RULES**

(rule 280 calculate-SN1 (calculate sn1 !j) -->

(setq diff (- (+ (+ (- (+ (\* Zr S0)(\* 9.36 (log (+ SN1 1) 10)))) 8.27)

(\* 2.32 (log Mr-base 10)))/(log (/ Delta-PSI 2.7) 10)

(+ 0.4 (/ 1094 (expt (+ SN1 1) 5.19)))) (log W18 10)))

(cond

((minusp diff)(setq sn1 (+ sn1 0.1)))

((> sn1 14) (db-delete '(calculate sn1 !j))

(format t" Please check your data for errors"))

```
(t (format t"
```

```
      Structural Number to protect the Base layer    = SN1 = ~d" sn1 )
(setf(get 'sn1 'value) sn1)
```

```
(setq sn2 1)
```

```
(db-insert '(calculate sn2 value ))
```

```
(db-delete '(calculate sn1 !j))))))
```

```
(rule 240 calculate-a1 (calculate a1 !j) -->
```

```
(terpri)(terpri)
```

```
(setq a1 (+ (* 0.4 (log (/ (get 'e-ac 'value ) 435000) 10)) 0.44))
```

```
(print"   The value of layer coefficient for AC    =a1 = ")
```

```
(prin1 a1)
```

```
(setf (get 'a1 'value)a1)
```

```
(db-insert '(calculate a2 value ))
```

```
(db-delete '(calculate a1 !j)))
```

```
(rule 200 calculate-D2* (calculate D2* !j) -->
```

```
(terpri)(terpri)
```

```
(setq D2 (/ (- (get 'sn2 'value) SN1*) (* a2 (get 'm2 'value))))
```

```
(setq D2* (round D2))
```

```
(setq SN2* (* a2 D2* m2))
```

```
(cond
```

```
((<= D2* 0)(terpri)(terpri)(format t"
```

The Base layer is not required. ")(setq D2\* 0)

(db-insert '(calculate D3\* value ))(db-delete '(calculate D2\* !j)))

((>= (+ SN1\* SN2\*) SN2)(terpri)(terpri)(format t"

Calculated thickness of the Base layer = D2 = ~d inches" D2)

(terpri)(format t"

Rounded thickness of the Base layer = D2\* = ~d inches " D2\*)

(terpri)(terpri)(terpri)(format t"

The actual combined Structural Number of the Surface layer and Base

course is greater than the Structural Number of the Sub-Base course i.e.

$(SN1* + SN2*) \geq SN2$  Or Numerically ,

$(\sim d + \sim d)(= \sim d) \geq \sim d \implies \text{OK}$  "

$SN1* SN2* (+ SN1* SN2*) SN2 )$

(terpri)(terpri)

(setf (get 'D2\* 'value)D2\*)

(db-insert '(calculate D3\* value ))

(db-delete '(calculate D2\* !j)))

(t(and(db-insert '(re-calculate D2\* value ))

(db-delete '(calculate D2\* !j))))))

;  
**INSERTING SOME PATTERNS INTO DATA BASE**

USER(1): (db-insert '(calculate D1\* value ))

(CALCULATE D1\* VALUE)

USER(2): (db-insert '(calculate SN1 value ))

(CALCULATE SN1 VALUE)

USER(3): (db-insert '(calculate a1 value ))

(CALCULATE A1 VALUE)

USER(4): (db-insert '(volume AC layer))

(VOLUME AC LAYER)

USER(5): (db-insert '(volume base layer ))

(VOLUME BASE LAYER)

;  
**GENERAL EXECUTION**

USER(6): (db-find '(calculate D1\* !j ))

((((( !j VALUE )))))

USER(7): (db-delete '(volume Base !j))

((VOLUME BASE LAYER))

USER(8): (tryrule calculate-a1)

THE VALUE OF LAYER COEFFICIENT FOR AC = A1 = 0.66

USER(9): (db-delete '(calculate a2 !j))

((CALCULATE A2 VALUE))

USER(10): (db-insert '(calculate a1 value ))

(CALCULATE A1 VALUE)

USER(11): (RUN)

Structural Number to protect the Base layer = SN1 = 3.5

The value of layer coefficient for AC = a1 = 0.66

NIL

USER(12):

USER(12): (db-find '(calculate D1\* !j ))

((((( !j VALUE )))))

USER(13): (db-find '(calculate SN1 !j ))

NIL

USER(14):





**Appendix-C**

;

**BASHROAD**

;

WELCOME TO "BASHROAD" FOR

;

;

;

DESIGNING OF ROADS

; This programme helps the engineer who is interested to design a road .It asks the user  
 ; about the specifications required for the road,the structural number required to protect  
 ; different layers of the flexible pavement, the properties of materials to be used in  
 ; different layers of the road, number of lanes required. Then keeping in view these  
 ; requirements, this programme estimates the thickness of different layers of the road,  
 ; quantity of the materials to be used and the total estimated cost of the road that can  
 ; meet the requirements.

;

**Start**

;

; This function is just to start the programme. It calls another function " bashroad ".

```
(defun Start ( )
  (bashroad))
```

;

**BashRoad**

;

; This function can also be used to start the programme. It calls another function  
 ; " no-of-lanes ".

```
(defun bashroad ( )
  (terpri)(terpri)(terpri)
  (format t "
```

## WELCOME TO 'BASHROAD' FOR

### DESIGNING OF ROADS

THIS PROGRAMME WILL GUIDE YOU TO ESTIMATE THE THICKNESS OF DIFFERENT LAYERS OF FLEXIBLE PAVEMENT/ SPECIFICATIONS OF A REQUIRED ROAD KEEPING IN VIEW YOUR REQUIREMENTS. IT CAN ALSO GIVE THE ESTIMATED QUANTITIES/COST OF DIFFERENT MATERIALS AND THE TOTAL COST OF THE ROAD (INCLUDING PER LANE-MILE COST). ")

(no-of-lanes))

#### No-of-lanes

;  
;

; This function is to ask user about required number of lanes. It ensures that the user  
; has entered a positive integer, puts this input into the data base and calls another  
; function " input-lane-width ".

```
(defun no-of-lanes ()
  (terpri)(terpri)(terpri)
  (print" PLEASE ENTER THE TOTAL NUMBER OF LANES REQUIRED FOR THE
ROAD (WRITE INTEGER ONLY).(EXAMPLE VALUE IS ' 4 ' ) ")(terpri)
  (let ((lanes (read)))
    (cond
      ((not(integerp lanes))(terpri)(terpri)(terpri)(print"YOU HAVE NOT ENTERED
        AN INTEGER ")(no-of-lanes))
      ((< lanes 1)(terpri)(terpri)(terpri)(print" THE MINIMUM NUMNBER
        OF LANES REQUIRED SHOULD BE '1' ")(no-of-lanes))
      (t(setf(get 'no-of 'lanes)lanes))))
  (setq lanes (get 'no-of 'lanes))(input-lane-width))
```

#### input-lane-width

;  
;

; This function is to ask user about required width of each lane. It ensures that the user  
; has entered a positive integer not less then specified limit, puts this input into the  
; data base and calls another function "input-road-length ".

```
(defun input-lane-width ()
  (terpri)(terpri)(terpri)
  (print" PLEASE ENTER WIDTH OF EACH LANE IN FEET. (WRITE INTEGER
ONLY). (TYPICAL VALUE IS ' 12 ' FEET) ")(terpri)
  (let ((width (read)))
```

```
(cond
  ((not(integerp width))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
    ENTERED AN INTEGER  ")(input-lane-width))
  ((< width 10)(terpri)(terpri)(terpri)(print" THE MINIMUM WIDTH
    OF LANE SHOULD BE ' 10 ' FEET  ")(input-lane-width))
  (t(setf(get 'width-of 'lane)width))))
(setq lane-width (get 'width-of 'lane))(input-road-length))
```

```
;
; Input-road-length
;
; This function is to ask user about required road length. It ensures that the user has
; entered a positive integer, puts this input into the data base and calls another function
; " sn-choice ".
```

```
(defun input-road-length ()
  (terpri)(terpri)(terpri)
  (print" PLEASE ENTER THE LENGTH OF THE REQUIRED ROAD IN MILES
    (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 10 ' )  ")(terpri)
  (let ((length (read)))
    (cond
      ((not(numberp length))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-road-length))
      ((= length 0)(terpri)(terpri)(terpri)(print"THE LENGTH OF THE REQUIRED
        ROAD SHOULD NOT BE ZERO  ")(input-road-length))
      ((< length 0)(terpri)(terpri)(terpri)(print"THE LENGTH OF THE
        REQUIRED ROAD SHOULD NOT BE A NEGATIVE NUMBER ")
        (input-road-length))
      (t(setf(get 'road 'length)length))))
  (setq length (get 'road 'length))(sn-choice))
```

```

;                                     sn-choice
;                                     =====
;
; This function gives the choice to user to enter the structural number required to
; protect base layer as data or user can ask that the system should calculate its value.
; It calls the function " input-sn1" or " input-years " accordingly. It can also
; exit the program if the user desires so.

```

```

(defun sn-choice ()
  (terpri)(terpri)(terpri)
  (format t" PLEASE ENTER A NUMBER FROM 1 TO 3.

      1. TO ENTER THE STRUCTURAL NUMBER REQUIRED TO PROTECT
      DIFFERENT LAYERS.
      2. BashRoad SHOULD CALCULATE THE STRUCTURAL NUMBER
      REQUIRED.
      3. TO QUIT BASH-ROAD.  ")(terpri)
  (let ((choice (read)))
    (cond
      ((equal choice 1)(setq calculate-SN-? 'No)(input-sn1))
      ((equal choice 2)(setq calculate-SN-? 'Yes)(input-years))
      ((equal choice 3)(format t" THANK YOU FOR USING BashRoad "))
      (t(format t" YOU HAVE NOT ENTERED A NUMBER FROM 1 TO 3. ")
        (sn-choice)(terpri)(terpri))))))

```

```

;                                     Input-years
;                                     =====
;
; This function is to ask user about the design period of the road in years. It ensures
; that the user has entered a number, puts this input into the data base and calls
; another function " input-ESAL ".

```

```

(defun input-years ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE DESIGN PERIOD IN YEARS (WRITE NUMBER
      ONLY).  EXAMPLE VALUE IS ' 10 ' YEARS.  ")(terpri)
  (let ((years (read)))
    (cond
      ((not(numberp years))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
      ENTERED A NUMBER")(input-years))
      (t(setf(get 'design 'period)years))))
  (setq years (get 'design 'period))(input-ESAL))

```

### **Input-ESAL**

```
;
;
; This function is to ask user about the equivalent single axel load (ESAL) for the
; design period of road. It ensures that the user has entered a positive number,
; puts this input into the data base and calls another function " input-DD ".
; If the user wants to give only initial year traffic, then it calls another function
; " input-ESALi ".
```

```
(defun input-ESAL ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE TWO WAY TRAFFIC I.E EQUIVALENT SINGLE
    AXEL LOAD (ESAL) FOR ~d YEARS IN MILLIONS. EXAMPLE VALUE
    FOR 10 YEARS IS ' 32 ' MILLIONS. IF YOU WANT TO GIVE INITIAL
    YEAR TRAFFIC, ENTER ' I ' ." years )(terpri)
  (let ((ESAL (read)))
    (cond
      ((equal ESAL 'I)(input-ESAL-i))
      ((not(numberp ESAL))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER OR 'I' ")(input-ESAL))
      ((<= ESAL 0)(terpri)(terpri)(terpri)(print"THE ESAL SHOULD NOT BE
        ZERO OR NEGATIVE NUMBER ")(input-ESAL))
      (t(setf(get 'total 'traffic) ESAL))))
    (setq ESAL (* 1000000 (get 'total 'traffic)))(input-DD))
```

### **Input-ESALi**

```
;
;
; This function is to ask user about the equivalent single axel load (ESAL) for the first
; year. It ensures that the user has entered a positive number, puts this input into the
; data base and calls another function " input-growth-rate ".
```

```
(defun input-ESAL-i ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE TWO WAY TRAFFIC (ESAL) FOR FIRST YEAR
    IN MILLIONS. EXAMPLE VALUE IS ' 2.67 ' MILLIONS. ")(terpri)
  (let ((ESAL-i (read)))
    (cond
      ((not(numberp ESAL-i))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-ESAL-i))
      ((<= ESAL-i 0)(terpri)(terpri)(terpri)(print"THE ESAL SHOULD NOT
        BE ZERO OR NEGATIVE ")(input-ESAL-i))
      (t(setf(get 'initial 'traffic) ESAL-i))))
    (setq ESAL-i (* 1000000 (get 'initial 'traffic)))(input-growth-rate))
```

### **Input-growth-rate**

```
;
;
; This function is to ask user about the annual growth rate of traffic. It ensures
; that the user has entered a number, puts this input into the data base and calls
; another function " Calculate-ESAL ".
```

```
(defun input-growth-rate ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE ANNUAL GROWTH RATE OF TRAFFIC IN
    PERCENT . EXAMPLE VALUE IS ' 4 ' PERCENT . ")(terpri)
  (let ((growth (read)))
    (cond
      ((not(numberp growth))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-growth-rate))
      (t(setf(get 'growth 'rate)growth))))
  (setq i (/ (get 'growth 'rate) 100)) (Calculate-ESAL))
```

### **Calculate-ESAL**

```
;
;
; This function is to calculate the equivalent single axel load (ESAL) for the design
; period of the road by using the values of equivalent single axel load (ESAL) for
; the first year, the annual growth rate of traffic and the design period of the road
; in years. It calls another function " input-DD ".
```

```
(defun Calculate-ESAL ()
  (setq ESAL (* ESAL-i (/ (- (expt (+ 1 i) years) 1) i)))
  (input-DD))
```

### **Input-DD**

```
;
;
; This function is to ask the user about the directional distribution factor for traffic.
; It ensures that the user has entered a number within specified range, puts this
; input into the data base and calls another function " input-LD ".
```



```
(defun input-DD ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE DIRECTIONAL DISTRIBUTION FACTOR.
    TYPICAL VALUE FOR TWO WAY TRAFFIC IS ' 0.5 ' . ")(terpri)
  (let ((DD (read)))
    (cond
      ((not(numberp DD))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-DD))
      ((> DD 1)(terpri)(terpri)(terpri)(format t"
        DIRECTIONAL DISTRIBUTION FACTOR CAN NOT BE GREATER
        THAN 1 ")
        (input-DD))
      (t(setf(get 'directional 'distribution) DD))))
  (setq DD (get 'directional 'distribution))(input-LD))
```

### **Input-LD**

```
;
;
; This function is to ask the user about the lane distribution factor for traffic.
; It ensures that the user has entered a number within specified range, puts this
; input into the data base and calls another function " Calculate-W18 ".
```

```
(defun input-LD ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE LANE DISTRIBUTION FACTOR FOR
    OUTER/DESIGN LANE. EXAMPLE VALUE IS ' 0.70 ' . ")(terpri)
  (let ((LD (read)))
    (cond
      ((not(numberp LD))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-LD))
      ((> LD 1)(terpri)(terpri)(terpri)(format t"
        LANE DISTRIBUTION FACTOR CAN NOT BE GREATER THAN 1 ")
        (input-LD))
      (t(setf(get 'lane 'distribution) LD))))
  (setq LD (get 'lane 'distribution))(Calculate-W18))
```

### **Calculate-W18**

```
;
;
; This function is to Calculate the total design/expected 18 kip load (W18) for the
; design period of the road by using the values of equivalent single axel load (ESAL),
; the directional distribution factor for traffic and the lane distribution factor for traffic.
; It calls another function " input-S0 ".
```

```
(defun Calculate-W18 ()
  (setq W18 (* (* ESAL DD) LD))
  (input-S0))
```

```
;
;                                     Input-S0
;                                     =====
; This function is to ask the user about the overall standard deviation. It ensures
; that the user has entered a number within specified range, puts this input into
; the data base and calls another function " input-reliability ".
```

```
(defun input-S0 ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE OVERALL STANDARD DEVIATION. EXAMPLE
    VALUE IS ' 0.45 ' . ")(terpri)
  (let ((S0 (read)))
    (cond
      ((not(numberp S0))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-S0))
      ((or(< S0 0.2)(> S0 0.6))(terpri)(terpri)(format t"
        OVERALL STANDARD DEVIATION SHOULD BE BETWEEN 0.2
        AND 0.6 ") (input-S0))
      (t(setf(get 'standard 'deviation) S0))))
  (setq S0 (get 'standard 'deviation))(input-reliability))
```

```
;
;                                     input-reliability
;                                     =====
; This function is to ask the user about the design reliability in percentage. It ensures
; that the user has entered a number within specified range, puts this input into the
; data base and calls another function " get-Zr-from-table ".
```

```
(defun input-reliability ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE DESIGN RELIABILITY IN PERCENT. TYPICAL
    VALUE IS ' 95 ' PERCENT . ")(terpri)
  (let ((R (read)))
    (cond
      ((not(numberp R))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-reliability))
      ((or(< R 50)(> R 99.99))(terpri)(terpri)(format t"
        DESIGN RELIABILITY SHOULD BE BETWEEN 50 AND 99.99
        PERCENT ")
```

```

      (input-reliability))
      (t(setf(get 'R 'value) R))))
(setq R (get 'R 'value))(get-Zr-from-table ))

```

```

;                                     Get-Zr-from-table
;
; This function is to get the value of the standard normal deviate (Zr) corresponding
; to the design reliability by using the table. It stores this value into the data base and
; calls another function " input-Delta-PSIo ".

```

```

(defun get-Zr-from-table ()
  (terpri)(terpri)
  (cond
    ((not(numberp R))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
      ENTERED A NUMBER")(input-reliability))
    ((or(< R 50)(> R 99.99))(terpri)(terpri)(format t"
      DESIGN RELIABILITY SHOULD BE BETWEEN 50 AND 99.99 PERCENT ")
      (input-reliability))
    ((= R 50) (setq Zr -0.000)(input-Delta-PSIo))
    ((<= R 60)(setq Zr -0.253)(input-Delta-PSIo))
    ((<= R 70)(setq Zr -0.524)(input-Delta-PSIo))
    ((<= R 75)(setq Zr -0.674)(input-Delta-PSIo))
    ((<= R 80)(setq Zr -0.841)(input-Delta-PSIo))
    ((<= R 85)(setq Zr -1.037)(input-Delta-PSIo))
    ((<= R 90)(setq Zr -1.282)(input-Delta-PSIo))
    ((<= R 91)(setq Zr -1.340)(input-Delta-PSIo))
    ((<= R 92)(setq Zr -1.405)(input-Delta-PSIo))
    ((<= R 93)(setq Zr -1.476)(input-Delta-PSIo))
    ((<= R 94)(setq Zr -1.555)(input-Delta-PSIo))
    ((<= R 95)(setq Zr -1.645)(input-Delta-PSIo))
    ((<= R 96)(setq Zr -1.751)(input-Delta-PSIo))
    ((<= R 97)(setq Zr -1.881)(input-Delta-PSIo))
    ((<= R 98)(setq Zr -2.054)(input-Delta-PSIo))
    ((<= R 99)(setq Zr -2.327)(input-Delta-PSIo))
    ((<= R 99.9)(setq Zr -3.090)(input-Delta-PSIo))
    ((<= R 99.99)(setq Zr -3.750)(input-Delta-PSIo))))

```

### **Input-Delta-PSIo**

```

;
;
; This function is to ask the user about the overall serviceability loss. It ensures
; that the user has entered a number within specified range, puts this input into
; the data base and calls another function " input-Delta-PSIenv ". If user wants
; to give the initial and terminal pavement serviceability index, then it calls another
; function " input-PSIi ".

```

```

(defun input-Delta-PSIo ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE OVERALL SERVICEABILITY LOSS .
  EXAMPLE VALUE IS ' 2 ' . IF YOU WANT TO GIVE INITIAL AND TERMINAL
  PAVEMENT SERVICEABILITY INDEX, ENTER ' I ' ." )(terpri)
  (let ((Delta-PSIo (read)))
    (cond
      ((equal Delta-PSIo 'I)(input-PSIi))
      ((not(numberp Delta-PSIo))(terpri)(terpri)(terpri)(print"YOU HAVE
      NOT ENTERED A NUMBER OR 'I' ")(input-Delta-PSIo ))
      ((or(< Delta-PSIo 1.2)(> Delta-PSIo 3.2))(terpri)(terpri)(format t"
      THE OVERALL SERVICEABILITY LOSS SHOULD BE BETWEEN 1.2
      AND 3.2 . ")
      (input-Delta-PSIo))
      (t(setf(get 'Delta-PSIo 'traffic) Delta-PSIo))))))
  (setq Delta-PSIo (get 'Delta-PSIo 'traffic))(input-Delta-PSIenv))

```

### **Input-PSIi**

```

;
;
; This function is to ask the user about the initial pavement serviceability index.
; It ensures that the user has entered a number within specified range, puts this
; input into the data base and calls another function " Input-PSIt ".

```

```

(defun input-PSIi ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE INITIAL PAVEMENT SERVICEABILITY INDEX.
  EXAMPLE VALUE IS ' 4.5 ' . ") (terpri)
  (let ((PSIi (read)))
    (cond
      ((not(numberp PSIi))(terpri)(terpri)(terpri)(print"YOU HAVE
      NOT ENTERED A NUMBER. ")(input-PSIi ))
      ((> PSIi 4.7) (terpri)(terpri)(format t"

```

```

INITIAL PAVEMENT SERVICEABILITY INDEX ABOVE 4.7 IS
IMPROBABLE .")
  (input-PSIi))
((< PSIi 1.5) (terpri)(terpri)(format t"
INITIAL PAVEMENT SERVICEABILITY INDEX BELOW 1.5 IS NOT
ADVISABLE AS THE ROAD WILL NOT BE SUITABLE FOR
TRAFFIC. ")(input-PSIi ))
(t(setf(get 'PSIi 'value) PSIi))))
(setq PSIi (get 'PSIi 'value))(input-PSIt))

```

### **Input-PSIt**

```

;
;
; This function is to ask the user about the terminal pavement serviceability
; index. It ensures that the user has entered a number within specified range,
; puts this input into the data base and calls another function " calculate-Delta-PSIo ".

```

```

(defun input-PSIt ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE TERMINAL PAVEMENT SERVICEABILITY
  INDEX. EXAMPLE VALUE IS ' 2.5 ' . ") (terpri)
  (let ((PSIt (read)))
    (cond
      ((not(numberp PSIt))(terpri)(terpri)(terpri)(print"YOU HAVE
      NOT ENTERED A NUMBER. ")(input-PSIt ))
      (> PSIt 4) (terpri)(terpri)(format t"
      TERMINAL PAVEMENT SERVICEABILITY INDEX ABOVE 4 IS NOT
      ADVISABLE BEEING VERY UNECHONOMICAL. ") (input-PSIt))
      (< PSIt 0) (terpri)(terpri)(format t"
      TERMINAL PAVEMENT SERVICEABILITY INDEX BELOW 0 IS NOT
      ADVISABLE AS THE ROAD WILL NOT BE PASSABLE FOR
      TRAFFIC. ")(input-PSIt ))
      (t(setf(get 'PSIt 'value) PSIt))))
  (setq PSIt (get 'PSIt 'value))(calculate-Delta-PSIo))

```

### **Calculate-Delta-PSIo**

```

;
;
; This function is to Calculate the overall serviceability loss by using the values
; of initial pavement serviceability index and terminal pavement serviceability index.
; It calls another function " input-Delta-PSIenv ".

```

```
(defun calculate-Delta-PSIo ()
  (setq Delta-PSIo (- PSli PSIt))
  (input-Delta-PSIenv))
```

```
;
; Input-Delta-PSIenv
;
; This function is to ask the user about the serviceability loss due to the environment.
; It ensures that the user has entered a number within specified range, puts this
; input into the data base and calls another function " calculate-Delta-PSI ".
```

```
(defun input-Delta-PSIenv ()
  (terpri)(terpri)
  (format t" PLEASE ENTER THE SERVICEABILITY LOSS DUE TO
    ENVIRONMENT. EXAMPLE VALUE IS ' 0.64 ' . ENTER ' 0 '(ZERO) TO
    IGNORE THE ENVIRONMENTAL EFFECTS ." )(terpri)
  (let ((Delta-PSIenv (read)))
    (cond
      ((not(numberp Delta-PSIenv))(terpri)(terpri)(print"YOU HAVE
        NOT ENTERED A NUMBER. ")(input-Delta-PSIenv ))
      ((or(< Delta-PSIenv 0)(> Delta-PSIenv 1.2))(terpri)(terpri)(format t"
        THE SERVICEABILITY LOSS DUE TO ENVIRONMENT SHOULD BE
        BETWEEN ' 0 ' AND ' 1.2' ")(input-Delta-PSIenv))
      (t(setf(get 'Delta-PSIenv 'value) Delta-PSIenv))))
  (setq Delta-PSIenv (get 'Delta-PSIenv 'value))(calculate-Delta-PSI))
```

```
;
; Calculate-Delta-PSI
;
; This function is to Calculate the design serviceability loss due to traffic, by
; using the values of overall serviceability loss and serviceability loss due to
; the environment. It calls another function " input-E-ac ".
```

```
(defun calculate-Delta-PSI ()
  (setq Delta-PSI (- Delta-PSIo Delta-PSIenv))
  (input-E-ac))
```

### **Input-sn1**

```
;
;
; This function is to ask user about the structural number required to protect base layer.
; It ensures that the user has entered a number within the specified range, puts this
; input into the data base and calls another function "input-sn2".
```

```
(defun input-sn1 ()
  (terpri)(terpri)(terpri)
  (print" PLEASE ENTER THE STRUCTURAL NUMBER REQUIRED TO PROTECT
    THE BASE (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 2.8 ' ")
    (terpri)
  (let ((sn1 (read)))
    (cond
      ((not(numberp sn1))(terpri)(terpri)(terpri)(print"YOU HAVE NOT ENTERED
        A NUMBER")(input-sn1))
      (t(setf(get 'sn1 'value)sn1))))
    (setq sn1 (get 'sn1 'value))(input-sn2))
```

### **Input-sn2**

```
;
;
; This function is to ask the user about the structural number required to protect
; Subbase layer. It ensures that the user has entered a number withinspecified range,
; puts this input into the data base and calls another function "input-sn3".
```

```
(defun input-sn2 ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE STRUCTURAL NUMBER REQUIRED TO PROTECT
    THE SUB BASE (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 3.8 ' ")
    (terpri)
  (let ((sn2 (read)))
    (cond
      ((not(numberp sn2))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-sn2))
      (t(setf(get 'sn2 'value)sn2))))
    (setq sn2 (get 'sn2 'value))(input-sn3))
```



### **Input-sn3**

```

;
;
; This function is to ask the user about structural number required to protect
; road bed soil. It ensures that the user has entered a number within specified range,
; puts this input into the data base and calls another function " input- E-ac ".

```

```

(defun input-sn3 ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE STRUCTURAL NUMBER REQUIRED TO PROTECT
    THE ROAD BED SOIL (WRITE NUMBER ONLY). (EXAMPLE VALUE
    IS ' 5.4 ' ") (terpri)
  (let ((sn3 (read)))
    (cond
      ((not(numberp sn3))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-sn3))
      (t(setf(get 'sn3 'value)sn3))))
    (setq sn3 (get 'sn3 'value))(input-E-ac))

```

### **Input-E-ac**

```

;
;
; This function is to ask user about Asphalt concrete modulus of elastisity. It ensures
; that user has entered a number within the specified range, puts this input into the
; data base and calls another function " input-Mr-base ".

```

```

(defun input-E-ac ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE ASPHALT CONCRETE MODULUS OF ELASTISITY
    IN PSI. (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 300,000 ' ")
    (terpri)
  (let ((E-ac (read)))
    (cond
      ((not(numberp E-ac))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-E-ac))
      ((or(> E-ac 500000)(< E-ac 50000))(terpri)(terpri)(terpri)
        (print" THE VALUE OF ASPHALT CONCRETE MODULUS OF
        ELASTISITY SHOULD BE BETWEEN ' 50,000 ' AND ' 500,000 ' psi ")
        (input-E-ac))
      (t(setf(get 'E-ac 'value)E-ac))))
    (setq E-ac (get 'E-ac 'value))(input-Mr-base))

```

### **Input-Mr-base**

```
;
;
; This function is to ask user about Granular Base resilient modulus. It ensures that
; user has entered a number within the specified range, puts this input into the
; data base and calls another function " input-Mr-Sub.base ".
```

```
(defun input-Mr-base ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE GRANULAR BASE RESILIENT MODULUS IN PSI.
    (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 25,000 ' ")(terpri)
  (let ((Mr-base (read)))
    (cond
      ((not(numberp Mr-base))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-Mr-base))
      ((or(> Mr-base 300000)(< Mr-base 15000))(terpri)(terpri)(terpri)
        (print" THE VALUE OF GRANULAR BASE RESILIENT MODULUS
          SHOULD BE BETWEEN ' 15,000 ' AND ' 300,000 ' psi ")
        (input-Mr-base))
      (t(setf(get 'Mr-base 'value)Mr-base))))
  (setq Mr-base (get 'Mr-base 'value))(input-Mr-sub.base))
```

### **Input-Mr-sub.base**

```
;
;
; This function is to ask user about SubBase resilient modulus. It ensures that user
; has entered a number within the specified range, puts this input into the data base
; and calls another function " Input-Mr-RoadBed ".
```

```
(defun input-Mr-sub.base ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE GRANULAR SUBBASE RESILIENT MODULUS IN
    PSI. (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 12,000 ' ")(terpri)
  (let ((Mr-sub.base (read)))
    (cond
      ((not(numberp Mr-sub.base))(terpri)(terpri)(terpri)(print"YOU HAVE
        NOT ENTERED A NUMBER")(input-Mr-sub.base))
      ((or(> Mr-sub.base 300000)(< Mr-sub.base 15000))
        (terpri)(terpri)(terpri)
        (print" THE VALUE OF GRANULAR SUBBASE RESILIENT
          MODULUS SHOULD BE BETWEEN ' 15,000 ' AND ' 300,000 ' psi ")
```

```

      (input-Mr-sub.base))
      (t(setf(get 'Mr-sub.base 'value)Mr-sub.base))))
    (setq Mr-sub.base (get 'Mr-sub.base 'value))(input-Mr-RoadBed))

```

### **Input-Mr-RoadBed**

```

;
;
; This function is to ask user about RoadBed soil resilient modulus. It ensures that
; user has entered a number, puts this input into the data base and calls another
; function " input-m2 ".

```

```

(defun input-Mr-RoadBed ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE ROAD BED SOIL RESILIENT MODULUS IN PSI.
  (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 2,200 ' ")(terpri)
  (let ((Mr-RoadBed (read)))
    (cond
      ((not(numberp Mr-RoadBed))(terpri)(terpri)(terpri)(print"YOU HAVE
      NOT ENTERED A NUMBER")(input-Mr-RoadBed))
      (t(setf(get 'Mr-RoadBed 'value)Mr-RoadBed))))
    (setq Mr-RoadBed (get 'Mr-RoadBed 'value))(input-m2))

```

### **Input-m2**

```

;
;
; This function is to ask user about drainage modifying factor /drainage coefficient
; for Base layer. It ensures that user has entered a number within the specified range,
; puts this input into the data base and calls another function " input-m3 ".

```

```

(defun input-m2 ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE DRAINAGE MODIFYING FACTOR FOR BASE
  LAYER . EXAMPLE VALUE IS ' 1.2 ' ")(terpri)
  (let ((m2 (read)))
    (cond
      ((not(numberp m2))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
      ENTERED A NUMBER")(input-m2))
      ((or(> m2 1.4)(< m2 0.6)) (terpri)(terpri)(terpri)
      (print" THE VALUE OF DRAINAGE MODIFYING FACTOR SHOULD
      BE BETWEEN ' 0.6 ' AND ' 1.4 ' ") (input-m2))
      (t(setf(get 'm2 'value)m2))))
    (setq m2 (get 'm2 'value))(input-m3))

```

### **Input-m3**

```

;
;
; This function is to ask user about drainage modifying factor /drainage coefficient
; for SubBase layer. It ensures that user has entered a number within the specified
; range, puts this input into the data base and calls another function " cost-calculation ".

```

```

(defun input-m3 ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE DRAINAGE MODIFYING FACTOR FOR SUB-BASE
    LAYER . EXAMPLE VALUE IS ' 0.7 ' ")(terpri)
  (let ((m3 (read)))
    (cond
      ((not(numberp m3))(terpri)(terpri)(terpri)(print"YOU HAVE NOT
        ENTERED A NUMBER")(input-m3))
      ((or(> m3 1.4)(< m3 0.6)) (terpri)(terpri)(terpri)
        (print" THE VALUE OF DRAINAGE MODIFYING FACTOR SHOULD
          BE BETWEEN ' 0.6 ' AND ' 1.4 ' ") (input-m3))
      (t(setf(get 'm3 'value)m3))))
  (setq m3 (get 'm3 'value))(cost-calculation))

```

### **Cost-calculation**

```

;
;
; This function is to ask user if he wants the material and cost estimation in addition
; to layer thicknesses. If yes, then it calls another function " input-AC-density "
; to get more data from the user. If no, then it calls another function " run " in order
; to fire the rules for road design (layer thicknesses) only, ignoring the rules for
; material and cost estimation .

```

```

(defun cost-calculation ()
  (terpri)(terpri)(terpri)
  (print" DO YOU WANT THE ESTIMATED QUANTITY AND COST OF
    MATERIL ALSO (IN ADDITION TO LAYER THICKNESSES ) 'Y' or 'N'  ")
  (terpri)
  (let ((cost (read)))
    (cond
      ((equal cost 'n)(if(equal calculate-SN-? 'Yes)(and (setq sn1 1)
        (db-insert '(start-from sn1 value ))(run))
        (and (db-insert '(start-from a1 value ))
          (setf(get 'cost 'required) 'no) (run))))
      ((equal cost 'y)(setf(get 'cost 'required) 'yes)(input-AC-density))))

```

### **Input-AC-density**

```
;
;
; This function is to ask user about the compacted density of Asphalt Concrete
; layer. It ensures that user has entered a number within the specified range, puts
; this input into the data base and calls another function "input-Base-density".
```

```
(defun input-AC-density ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE COMPACTED DENSITY OF ASPHALT CONCRETE
    LAYER (IN Lbs/cu-ft)(WRITE NUMBER ONLY).(TYPICAL VALUE
    IS ' 150 ' Lbs/cu-ft) ") (terpri)
  (let ((ac-density (read)))
    (cond
      ((not(numberp ac-density))(terpri)(terpri)(terpri)(print"YOU HAVE
        NOT ENTERED A NUMBER")(input-AC-density))
      ((or(> ac-density 160)(< ac-density 135))(terpri)(terpri)(terpri)
        (print" THE COMPACTED DENSITY OF ASPHALT CONCRETE
          SHOULD BE BETWEEN ' 135 ' AND ' 160 ' pcf ")
        (input-AC-density))
      (t(setf(get 'AC-density 'value)AC-density))))
  (setq AC-density (get 'AC-density 'value))(input-base-density))
```

### **Input-Base-density**

```
;
;
; This function is to ask user about the compacted density of Base layer. It ensures
; that user has entered a number within the specified range, puts this input into the
; data base and calls another function "input-SubBase-density".
```

```
(defun input-Base-density ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE COMPACTED DENSITY OF BASE LAYER (IN
    Lbs/cu-ft)(WRITE NUMBER ONLY).(TYPICAL VALUE IS ' 135 ' Lbs/cu-ft) ")
  (terpri)
  (let ((Base-density (read)))
    (cond
      ((not(numberp Base-density))(terpri)(terpri)(terpri)
        (print"YOU HAVE NOT ENTERED A NUMBER")(input-Base-density))
      ((or(> Base-density 140)(< Base-density 115))(terpri)(terpri)
```

```

(terpri) (print" THE COMPACTED DENSITY OF BASE LAYER
          SHOULD BE BETWEEN ' 115 ' AND ' 140 ' pcf ")
          (input-Base-density))
(t(setf(get 'Base-density 'value)Base-density))))
(setq Base-density (get 'Base-density 'value))(input-Subbase-density))

```

```

;                                     Input-SubBase-density
;                                     =====
;
; This function is to ask user about the compacted density of SubBase layer.
; It ensures that user has entered a number within the specified range, puts this input
; into the data base and calls another function " input-Ac-cost ".

(defun input-SubBase-density ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE COMPACTED DENSITY OF SUB-BASE LAYER (IN
        Lbs/cu-ft) (WRITE NUMBER ONLY).(TYPICAL VALUE IS ' 125 ' Lbs/cu-ft)")
    (terpri)
  (let ((SubBase-density (read)))
    (cond
      ((not(numberp SubBase-density))(terpri)(terpri)(terpri)
        (print"YOU HAVE NOT ENTERED A NUMBER")(input-SubBase-density))
      ((or(> SubBase-density 140)(< SubBase-density 110))(terpri)(terpri)
        (terpri) (print" THE COMPACTED DENSITY OF SUB-BASE LAYER
          SHOULD BE BETWEEN ' 110 ' AND ' 140 ' pcf ")
          (input-SubBase-density))
        (t(setf(get 'SubBase-density 'value)SubBase-density))))
    (setq SubBase-density (get 'SubBase-density 'value))(input-Ac-cost))

```

```

;                                     Input-Ac-cost
;                                     =====
;
; This function is to ask user about cost of the Asphalt Concrete mix. It ensures
; that user has entered a number within the specified range, puts this input into the
; data base and calls another function "input-Base-cost ".

```

```

(defun input-Ac-cost ()
  (terpri)(terpri)(terpri)
  (format t"PLEASE ENTER THE COST OF ASPHALT CONCRETE MATERIAL IN
          DOLLARS/TON (WRITE NUMBER ONLY).(TYPICAL RANGE IS
          BETWEEN ' 15 ' AND ' 50 ' DOLLARS/TON) ")(terpri)

```

```
(let ((Ac-cost (read)))
  (cond
    ((not(numberp Ac-cost))(terpri)(terpri)(terpri)
      (print"YOU HAVE NOT ENTERED A NUMBER")(input-Ac-cost))
    (t(setf(get 'Ac-cost 'value)Ac-cost))))
(setq Ac-cost (get 'Ac-cost 'value))(input-Base-cost))
```

### **Input-Base-cost**

```
;
;
; This function is to ask user about cost of the Base material. It ensures that user has
; entered a number within the specified range, puts this input into the data base
; and calls another function " input-SubBase-cost ".
```

```
(defun input-Base-cost ()
  (terpri)(terpri)(terpri)
  (format t"PLEASE ENTER THE COST OF BASE MATERIAL IN DOLLARS/TON
    (WRITE NUMBER ONLY).(TYPICAL RANGE IS BETWEEN ' 8 ' AND ' 25 '
    DOLLARS/TON) ")(terpri)
  (let ((Base-cost (read)))
    (cond
      ((not(numberp Base-cost ))(terpri)(terpri)(terpri)
        (print"YOU HAVE NOT ENTERED A NUMBER")(input-Base-cost))
      (t(setf(get 'Base-cost 'value)Base-cost))))
  (setq Base-cost (get 'Base-cost 'value))(input-SubBase-cost))
```

### **Input-SubBase-cost**

```
;
;
; This function is to ask user about cost of the SubBase material. It ensures that user
; has entered a number within the specified range, puts this input into the data base
; and calls another function " input-Overhead-charges ".
```

```
(defun input-SubBase-cost ()
  (terpri)(terpri)(terpri)
  (format t"PLEASE ENTER THE COST OF SUB-BASE MATERIAL IN
    DOLLARS/TON (WRITE NUMBER ONLY).(TYPICAL RANGE IS
    BETWEEN ' 3 ' AND ' 12 ' DOLLARS/TON) ")(terpri)
  (let ((SubBase-cost (read)))
    (cond
```

```

((not(numberp SubBase-cost ))(terpri)(terpri)(terpri)
 (print"YOU HAVE NOT ENTERED A NUMBER")(input-SubBase-cost))
(t(setf(get 'SubBase-cost 'value)SubBase-cost))))
(setq SubBase-cost (get 'SubBase-cost 'value))(input-Overhead-charges))

```

### **Input-Overhead-charges**

```

;
;
;
; This function is to ask user about the Overhead-charges for one mile of each lane .
; (It includes the labour charges, Equipment depreciation and other miscelanious
; charges. It ensures that user has entered a number within the specified range,
; puts this input into the data base and calls another function " Run " in order to
; fire the rules for road design (layer thicknesses), material estimation and cost
; estimation.

```

```

(defun input-Overhead-charges ()
  (terpri)(terpri)(terpri)
  (print"PLEASE ENTER THE OVERHEAD CHARGES FOR 1 MILE OF EACH
    LANE(IN DOLLARS/LANE/MILE) (WRITE NUMBER ONLY).(EXAMPLE
    VALUE IS ' 7000 ' DOLLARS /LANE-MILE) ")(terpri)
  (let ((Overhead-cost (read)))
    (cond
      ((not(numberp Overhead-cost ))(terpri)(terpri)(terpri)
        (print"YOU HAVE NOT ENTERED A NUMBER")(input-Overhead-charges))
      (t(setf(get 'Overhead-plm 'value ) Overhead-cost))))
  (setq Overhead-plm (get 'Overhead-plm 'value))
  (setq Total-Overhead-pm (* Overhead-plm lanes))
  (setq Total-Overhead (* Total-Overhead-pm length))
  (cond
    ((equal calculate-SN-? 'Yes)
      (setq sn1 1)(db-insert '(start-from sn1 value ))(run))
    ((equal calculate-SN-? 'No) (db-insert '(start-from a1 value ))(run))
    (t(format t "Error in function 'sn-choice'. "))))

```



```
;INSERT A MINI-DATABASE
```

```
(mapcar #'db-insert '(
(road overhead charges)
(lanes -- --)
))
```

```
::
::
```

## **RULES**

```
::
::
```

### **Start-from-SN1**

```
:: This Rule fires at its Priority(290),if the pattern (start-from sn1 !j) matches. This
;; pattern is only inserted if the user wants that SN values should be calculated
;; within the system. It gives the indication that the results are beeing printed.
;; It inserts the pattern to fire the rule "calculate-sn1" and deletes its own pattern.
```

```
(rule 290 start-from-SN1 (start-from SN1 !j) -->
  (terpri)(terpri)(terpri)(format t "
```

---

---

## **DESIGN RESULTS**

---

---

```
"
```

```
(db-insert '(calculate sn1 value ))
(db-delete '(start-from sn1 !j)))
```

```
::
::
```

### **Calculate-SN1**

```
:: This Rule fires at its Priority(280), if the pattern (calculate sn1 !j) matches. It calculates
;; the value of Structural Number to protect the Base layer (sn1) by using the value of
;; the standard normal deviate (Zr) corresponding to the design reliability, overall
;; standard deviation(S0), Granular Base resilient modulus, design serviceability loss
```

;; due to traffic and total design/expected 18 kip load (W18) for the design period.  
 ;; It inserts the pattern to fire the rule "calculate-sn2" and deletes its own pattern.

```
(rule 280 calculate-SN1 (calculate sn1 !j) -->
  (setq diff (- (+ (+ (- (+ (* Zr S0)(* 9.36 (log (+ SN1 1) 10))) 8.27)
    (* 2.32 (log Mr-base 10)))/ (log (/ Delta-PSI 2.7) 10)
    (+ 0.4 (/ 1094 (expt (+ SN1 1) 5.19))))) (log W18 10)))
  (cond
    ((minusp diff)(setq sn1 (+ sn1 0.1)))
    ((> sn1 14)(db-delete '(calculate sn1 !j))
      (format t " Please check your data for errors"))
    (t(format t "
```

```
Structural Number to protect the Base layer    = SN1 = ~d" sn1 )
  (setf(get 'sn1 'value) sn1)
  (setq sn2 1)
  (db-insert '(calculate sn2 value ))
  (db-delete '(calculate sn1 !j))))
```

```
;;
;;
;;
```

### **Calculate-SN2**

;; This Rule fires at its Priority(270), if the pattern (calculate sn2 !j) matches. It calculates  
 ;; the value of Structural Number to protect the SubBase layer (sn2) by using the value  
 ;; of the standard normal deviate (Zr) corresponding to the design reliability, overall  
 ;; standard deviation(S0), SubBase resilient modulus, design serviceability loss due to  
 ;; traffic and total design/expected 18 kip load (W18) for the design period. It  
 ;; inserts the pattern to fire the rule "calculate-sn3" and deletes its own pattern.

```
(rule 270 calculate-SN2 (calculate sn2 !j) -->
  (setq diff (- (+ (+ (- (+ (* Zr S0)(* 9.36 (log (+ SN2 1) 10))) 8.27)
    (* 2.32 (log Mr-sub.base 10)))/ (log (/ Delta-PSI 2.7) 10)
    (+ 0.4 (/ 1094 (expt (+ SN2 1) 5.19))))) (log W18 10)))
  (cond
    ((minusp diff)(setq sn2 (+ sn2 0.1)))
    ((> sn2 14)(db-delete '(calculate sn1 !j))
      (format t " Please check your data for errors"))
    (t(format t "
```

```
Structural Number to protect the Sub-Base layer = SN2 = ~d " sn2 )
  (setf(get 'sn2 'value) sn2)
  (setq sn3 1)
  (db-insert '(calculate sn3 value ))
  (db-delete '(calculate sn2 !j))))
```

### **Calculate-SN3**

---

```
;;
;;
;; This Rule fires at its Priority(260), if the pattern (calculate sn3 !j) matches. It calculates
;; the value of Structural Number to protect the RoadBed soil(sn2) by using the value
;; of the standard normal deviate (Zr) corresponding to the design reliability, overall
;; standard deviation(S0), RoadBed soil resilient modulus, design serviceability loss
;; due to traffic and total design/expected 18 kip load (W18) for the design period. It
;; inserts the pattern to fire the rule "start-from-a1" and deletes its own pattern.
```

```
(rule 260 calculate-SN3 (calculate sn3 !j) -->
  (setq diff (- (+ (+ (- (+ (* Zr S0) (* 9.36 (log (+ SN3 1) 10))) 8.27)
    (* 2.32 (log Mr-RoadBed 10)))/ (log (/ Delta-PSI 2.7) 10)
    (+ 0.4 (/ 1094 (expt (+ SN3 1) 5.19))))) (log W18 10)))
  (cond
    ((minusp diff)(setq sn3 (+ sn3 0.1)))
    ((> sn3 14)(db-delete '(calculate sn1 !j))
      (format t " Please check your data for errors"))
    (t(format t "
```

```
Structural Number to protect the Road Bed Soil = SN3 = ~d " sn3 )
  (setf(get 'sn3 'value) sn3)
  (db-insert '(calculate a1 value ))
  (db-delete '(calculate sn3 !j))))
```

### **Start-from-a1**

---

```
;;
;;
;; This Rule fires at its Priority(250),if the pattern (start-from a1 !j) matches. This
;; pattern is only inserted if the user wants to give SN values as input, which may have
;; been calculated outside this program. It gives the indication that the
;; results are following. It inserts the pattern to fire the rule "calculate-a1" and
;; deletes its own pattern.
```

```
(rule 250 start-from-a1 (start-from a1 !j) -->
  (terpri)(terpri)(terpri)(terpri)(format t "
```

---



---

## DESIGN RESULTS

---



---

)

```
(db-insert '(calculate a1 value ))
(db-delete '(start-from a1 !j)))
```

```
::
::
::
```

### Calculate-a1

```
:: This Rule fires at its Priority(240), if the pattern (calculate a1 !j) matches. It calculates
;; the value of layer coefficient for AC (a1) by using Asphalt concrete modulus of
;; elasticity. It inserts the pattern to fire the rule "calculate-a2" and deletes its own
;; pattern.
```

```
(rule 240 calculate-a1 (calculate a1 !j) -->
  (terpri)(terpri)
  (setq a1 (+ (* 0.4 (log (/ (get 'e-ac 'value ) 435000) 10)) 0.44))
  (print" The value of layer coefficient for AC    =a1 = ")
  (prin1 a1)
  (setf (get 'a1 'value)a1)
  (db-insert '(calculate a2 value ))
  (db-delete '(calculate a1 !j)))
```

```
::
::
::
```

### Calculate-a2

```
:: This Rule fires at its Priority (230), if the pattern (calculate a2 !j) matches. It calculates
;; the value of layer coefficient for Base (a2) by using Granular Base resilient modulus.
;; It inserts the pattern to fire the rule "calculate-a3" and deletes its own pattern.
```

```
(rule 230 calculate-a2 (calculate a2 !j) -->
  (terpri)(terpri)
  (setq a2 (- (* 0.249 (log (get 'Mr-base 'value) 10)) 0.977))
  (print" The value of layer coefficient for BASE    =a2 = ")
  (prin1 a2)
  (setf (get 'a2 'value)a2)
  (db-insert '(calculate a3 value ))
  (db-delete '(calculate a2 !j)))
```

```

;;
;; Calculate-a3
;;
;; This Rule fires at its Priority (220), if the pattern (calculate a3 !j) matches. It calculates
;; the value of layer coefficient for SubBase (a3) by using SubBase resilient modulus.
;; It inserts the pattern to fire the rule "calculate-D1*" and deletes its own pattern.

```

```

(rule 220 calculate-a3 (calculate a3 !j) -->
  (terpri)(terpri)
  (setq a3 (- (* 0.227 (log (get 'Mr-sub.base 'value) 10)) 0.839))
  (print" The value of layer coefficient for SUB-BASE =a3 = ")
  (prin1 a3)
  (setf (get 'a3 'value)a3)
  (db-insert '(calculate D1* value))
  (db-delete '(calculate a3 !j)))

```

```

;;
;; Calculate-D1*
;;
;; This Rule fires at its Priority (210), if the pattern (calculate D1* !j) matches. It
;; calculates the thickness of the AC layer(D1) by using structural number required
;; to protect base layer(sn1) and layer coefficient for AC(a1). It rounds up the thickness
;; of the AC layer to get D1* and then calculates the actual Structural Number for AC
;; layer (SN1*). It inserts the pattern to fire the rule "calculate-D2*" and deletes its
;; own pattern.

```

```

(rule 210 calculate-D1* (calculate D1* !j) -->
  (terpri)(terpri)
  (setq D1 (/ (get 'sn1 'value) a1))
  (format t"
Calculated thickness of the AC layer    = D1 = ~d inches" D1)
  (terpri)(terpri)
  (setq D1* (fceiling D1))
  (format t"
Rounded-Up thickness of the AC layer    = D1* = ~d inches " D1*)
  (terpri)(terpri)
  (setq SN1* (* a1 D1*))
  (setq SN1 (get 'sn1 'value))
  (format t"
The actual Structural Number for AC layer = SN1* = ~d >= SN1(~d) ==> OK      "
SN1* SN1) (terpri)
  (setf (get 'D1* 'value)D1*)
  (db-insert '(calculate D2* value ))
  (db-delete '(calculate D1* !j)))

```

```

;;
;; Calculate-D2*
;;
;; This Rule fires at its Priority (200), if the pattern (calculate D2* !j) matches. It
;; calculates the thickness of the Base layer(D2) by using structural number required
;; to protect Subbase layer(sn2), SN1* , layer coefficient for Base(a2) and drainage
;; coefficient for Base layer. It rounds up/down the thickness of the Base layer to get
;; D2* and then calculates the actual Structural Number for Base layer (SN2*). It also
;; ensures that the conditions are satisfied. It inserts the pattern to fire the rule
;; "calculate-D3*"/ "re-calculate-D2*") and deletes its own pattern.

```

```

(rule 200 calculate-D2* (calculate D2* !j) -->
  (terpri)(terpri)
  (setq D2 (/ (- (get 'sn2 'value) SN1*) (* a2 (get 'm2 'value))))
  (setq D2* (round D2))
  (setq SN2* (* a2 D2* m2))
  (cond
    ((<= D2* 0)(terpri)(terpri)(format t"
The Base layer is not required. ")(setq D3* 0)
  (db-insert '(calculate D3* value ))(db-delete '(calculate D2* !j)))
    ((>= (+ SN1* SN2*) SN2)(terpri)(terpri)(format t"
Calculated thickness of the Base layer = D2 = ~d inches" D2)(terpri)
  (terpri)(format t"
Rounded thickness of the Base layer = D2* = ~d inches " D2*)
  (terpri)(terpri)(terpri)(format t"
The actual combined Structural Number of the Surface layer and Base
course is greater than the Structural Number of the Sub-Base course i.e

```

(SN1\* + SN2\*) >= SN2      Or Numerically ,

```

( ~d + ~d )( = ~d ) >= ~d ==> OK      "
SN1* SN2* (+ SN1* SN2*) SN2 )
(terpri)(terpri)
(setf (get 'D2* 'value)D2*)
(db-insert '(calculate D3* value ))
(db-delete '(calculate D2* !j)))
(t(and(db-insert '(re-calculate D2* value ))
  (db-delete '(calculate D2* !j))))))

```

;; **Re-calculate-D2\***  
 ;;  
 ;; This Rule fires at its Priority (190), if the pattern (re-calculate D2\* !j) matches.  
 ;; It increases the thickness of the Base layer(D2\*) if the conditions are not satisfied.  
 ;; It inserts the pattern to fire the rule "calculate-D3\*"/ "again-re-calculate-D2\*) and  
 ;; deletes its own pattern.

(rule 190 re-calculate-D2\* (re-calculate D2\* !j) -->  
 (terpri)(terpri) (format t"  
 Calculated thickness of the Base layer = D2 = ~d inches" D2)(terpri)  
 (terpri)(format t"  
 Rounded thickness of the Base layer = D2\* = ~d inches " D2\*)  
 (terpri) (terpri)(terpri)(format t"  
 The actual combined Structural Number of the Surface layer and Base  
 course is not greater than the Structural Number of the Sub-Base course  
 i.e  $(SN1* + SN2*) < SN2$  Or Numerically ,  
 $(\sim d + \sim d)(= \sim d) < \sim d \implies \text{Incorrect}$

So increasing the thickness of the Base layer by 1 inch "  
 $SN1* SN2* (+ SN1* SN2*) SN2$  ) (terpri)  
 (setq D2\* (+ D2\* 1))  
 (setq SN2\* (\* a2 D2\* m2))  
 (cond  
 ((>= (+ SN1\* SN2\*) SN2)(terpri)(terpri)(format t"  
 Now increased thickness of the Base layer = D2\* = ~d inches " D2\*)  
 (terpri)(terpri)(terpri)(format t"  
 The actual combined Structural Number of the Surface layer and Base  
 course is greater than the Structural Number of the Sub-Base course i.e

$(SN1* + SN2*) \geq SN2$  Or Numerically ,  
 $(\sim d + \sim d)(= \sim d) \geq \sim d \implies \text{OK}$  "  
 $SN1* SN2* (+ SN1* SN2*) SN2$  )  
 (terpri)(terpri)  
 (setf (get 'D2\* 'value)D2\*)  
 (db-insert '(calculate D3\* value ))  
 (db-delete '(re-calculate D2\* !j)))  
 (t(and(db-insert '(again-re-calculate D2\* value ))  
 (db-delete '(re-calculate D2\* !j))))))

```

;;                                     again-re-calculate-D2*
;;                                     again-re-calculate-D2*
;;
;; This Rule fires at its Priority (180), if the pattern (again-re-calculate D2* !j) matches.
;; It increases the thickness of the Base layer(D2*) if the conditions are not satisfied.
;; It inserts the pattern to fire the rule "calculate-D3*"/ "again-re-calculate-D2*) and
;; deletes its own pattern.

```

```

(rule 180 again-re-calculate-D2* (again-re-calculate D2* !j) -->
  (terpri)(terpri) (terpri)(format t"
    Now increased thickness of the Base layer   = D2* = ~d inches " D2*)
    (terpri)(terpri)(terpri)(format t"
    But still the actual combined Structural Number of the Surface layer
    and Base course is less than the Structural Number of the Sub-Base
    course   i.e   (SN1* + SN2*) < SN2      Or Numerically ,

```

( ~d + ~d )( = ~d ) < ~d ==> Incorrect

So again increasing the thickness of the Base layer by 1 inch "

```

  SN1* SN2* (+ SN1* SN2*) SN2 )
  (terpri)(terpri)
  (setq D2* (+ D2* 1))
  (setq SN2* (* a2 D2* m2))
  (cond
    ((>= (+ SN1* SN2*) SN2)(terpri)(terpri) (terpri)(format t"
    Now increased thickness of the Base layer   = D2* = ~d inches " D2*)
    (terpri)(terpri)(terpri)(format t"
    Now the actual combined Structural Number of the Surface layer and Base
    course is greater than the Structural Number of the Sub-Base course i.e.

```

(SN1\* + SN2\*) >= SN2 Or Numerically ,

( ~d + ~d )( = ~d ) >= ~d ==> OK "

```

  SN1* SN2* (+ SN1* SN2*) SN2 )
  (terpri)(terpri)
  (setf (get 'D2* 'value)D2*)
  (db-insert '(calculate D3* value ))
  (db-delete '(again-re-calculate D2* !j))))))

```



**Calculate-D3\***

---

```
;;
;;
;; This Rule fires at its Priority (170), if the pattern (calculate D3* !j) matches.
;; It calculates the thickness of the SubBase layer(D3) by using structural number
;; required to protect Road bed soil(sn3), SN1* , SN2* , layer coefficient for
;; subBase (a3) and drainage coefficient for subBase layer(m3). It rounds up/down
;; the thickness of the Base layer to get D3* and then calculates the actual Structural
;; Number for SubBase layer (SN3*). It also ensures that the conditions are satisfied.
;; It inserts the pattern to fire the rule "road-cost"/ "re-calculate-D3*) and deletes its
;; own pattern.
```

```
(rule 170 calculate-D3* (calculate D3* !j) -->
  (terpri)(terpri)
  (setq D3 (/ (- SN3 SN1* SN2*) (* a3 m3)))
  (setq D3* (round D3))
  (setq SN3* (* a3 D3* m3))
  (cond
    ((<= D3* 0)(terpri)(terpri)(format t"
The SubBase layer is not required. ")(setq D3* 0)
  (db-insert '(road cost required ))(db-delete '(calculate D3* !j)))
    ((>= (+ SN1* SN2* SN3*) SN3)(terpri)(terpri)(format t"
Calculated thickness of the SubBase layer = D3 = ~d inches" D3)(terpri)
  (terpri)(format t"
Rounded thickness of the SubBase layer = D3* = ~d inches " D3*)
  (terpri)(terpri)(terpri)(format t"
The actual combined Structural Number of the Surface layer and Base and
SubBase is greater than the Structural Number of the Sub-Base course
```

i.e  $(SN1* + SN2* + SN3*) \geq SN3$  Or Numerically ,

```
( ~d + ~d + ~d)( = ~d ) >= ~d ==> OK      "
SN1* SN2* SN3* (+ SN1* SN2* SN3*) SN3 )
(terpri)(terpri)
(setf (get 'D3* 'value)D3*)
(db-insert '(road cost required))
(db-delete '(calculate D3* !j)))
(t(and(db-insert '(re-calculate D3* value ))
  (db-delete '(calculate D3* !j)))))
```

;; **Re-calculate-D3\***  
 ;;  
 ;;  
 ;; This Rule fires at its Priority (160), if the pattern (re-calculate D3\* !j) matches.  
 ;; It increases the thickness of the SubBase layer(D3\*) if the conditions are not satisfied.  
 ;; It inserts the pattern to fire the rule "road-cost"/ "again-re-calculate-D3\*) and  
 ;; deletes its own pattern.

```
(rule 160 re-calculate-D3* (re-calculate D3* !j) -->
  (terpri)(terpri) (format t"
    Calculated thickness of the SubBase layer = D3 = ~d inches" D3)(terpri)
    (terpri)(format t"
    Rounded thickness of the SubBase layer = D3* = ~d inches " D3*)
    (terpri) (terpri)(terpri)(format t"
```

The actual combined Structural Number of the Surface layer , Base and SubBase is not greater than the Structural Number of the RoadBed soil.

i.e  $(SN1* + SN2* + SN3*) < SN3$  Or Numerically,

$(\sim d + \sim d + \sim d)(= \sim d) < \sim d \implies \text{Incorrect}$

So increasing the thickness of the SubBase layer by 1 inch "

```
SN1* SN2* SN3*(+ SN1* SN2* SN3*) SN3 ) (terpri)
(setq D3* (+ D3* 1))
(setq SN3* (* a3 D3* m3))
(cond
  ((>= (+ SN1* SN2* SN3*) SN3)(terpri)(terpri)(format t"
    Now increased thickness of the SubBase layer = D3* = ~d inches " D3*)
    (terpri)(terpri)(terpri)(format t"
```

The actual combined Structural Number of the Surface layer, Base and

SubBase is greater than the Structural Number of the RoadBed soil. i.e.

$(SN1* + SN2* + SN3*) \geq SN3$  Or Numerically,

```
( ~d + ~d + ~d )(= ~d ) >= ~d ==> OK      "
SN1* SN2* SN3*(+ SN1* SN2* SN3*) SN3 )
(terpri)(terpri)
(setf (get 'D3* 'value)D3*)
(db-insert '(road cost required ))
(db-delete '(re-calculate D3* !j)))
(t(and(db-insert '(again-re-calculate D3* value ))
  (db-delete '(re-calculate D3* !j))))))
```

```

;;                                     again-re-calculate-D3*
;;                                     =====
;; This Rule fires at its Priority (150), if the pattern (again-re-calculate D3* !j) matches.
;; It increases the thickness of the SubBase layer(D3*) if the conditions are not satisfied.
;; It inserts the pattern to fire the rule "road-cost" and deletes its own pattern.

```

```

(rule 150 again-re-calculate-D3* (again-re-calculate D3* !j) -->
  (terpri)(terpri) (terpri)(format t"
    Now increased thickness of the SubBase layer  = D3* = ~d inches " D3*)
  (terpri)(terpri)(terpri)(format t"

```

But still the actual combined Structural Number of the Surface layer,  
Base and SubBase is less than the Structural Number of the RoadBed soil.

i.e  $(SN1* + SN2* + SN3*) < SN3$  Or Numerically,

$(\sim d + \sim d + \sim d)(= \sim d) < \sim d \implies \text{Incorrect}$

So again increasing the thickness of the SubBase layer by 1 inch "

```

  SN1* SN2* SN3*(+ SN1* SN2* SN3*) SN3 )
  (terpri)(terpri)
  (setq D3* (+ D3* 1))
  (setq SN3* (* a3 D3* m3))
  (cond
    ((>= (+ SN1* SN2* SN3*) SN3)(terpri)(terpri) (terpri)(format t"
    Now increased thickness of the SubBase layer  = D3* = ~d inches " D3*)
    (terpri)(terpri)(terpri)(format t"
    Now the actual combined Structural Number of the Surface layer, Base

```

and SubBase is greater than the Structural Number of the RoadBed soil.

i.e  $(SN1* + SN2* + SN3*) \geq SN3$  Or Numerically,

$(\sim d + \sim d + \sim d)(= \sim d) \geq \sim d \implies \text{OK}$  "

```

  SN1* SN2* SN3*(+ SN1* SN2* SN3*) SN3 )
  (terpri)(terpri)
  (setf (get 'D3* 'value)D3*)
  (db-insert '(road cost required))
  (db-delete '(again-re-calculate D3* !j))))

```

```

;;                                     road-cost
;;                                     =====
;; This Rule fires at its Priority (140), if the pattern (road cost !j) matches. If user wants
;; cost estimation also, it inserts the pattern to fire the rule "AC-volume", otherwise
;; it ends the program. Deletes its own pattern also.

```

```

(rule 140 road-cost (road cost !j) -->
  (terpri)(terpri)
  (cond
    ((equal (get 'cost 'required) 'no)
      (db-delete '(road cost !j))
      (format t"

----- THE END ----- ")
      (halt))
    ((equal (get 'cost 'required) 'yes)
      (db-insert '(volume AC layer))
      (db-delete '(road cost !j))))))

```

```

;;                                     AC-volume
;;                                     =====
;; This Rule fires at its Priority(130), if the pattern (volume AC !j) matches. It calculates
;; the total compacted Volume of Asphalt Concrete Mix required by using required
;; number of lanes, width of each lane, road length and D1*. It inserts pattern to fire the
;; rule "Base-volume" and deletes its own pattern.

```

```

(rule 130 AC-volume (volume AC !j) -->
  (terpri)(terpri)
  (setq AC-v (* (* (* lanes lane-width) (* length 5280 ))
    (/ D1 * 12)))

  (format t"
The total compacted Volume of Asphalt Concrete Mix required = ~d Cu-ft
" AC-v )
  (terpri)(terpri)
  (db-insert '(volume Base layer))
  (db-delete '(volume AC !j)))

```

```

;;                                     Base-volume
;;                                     =====
;; This Rule fires at its Priority(120), if the pattern(volume Base !j) matches. It calculates
;; the total compacted Volume of Base material required by using required number of
;; lanes, width of each lane, road length and D2*. It inserts pattern to fire the rule
;; "SubBase-volume" and deletes its own pattern.

```

```

(rule 120 Base-volume (volume Base !j) -->
  (terpri)(terpri)
  (setq B-v (* (* (* lanes lane-width) (* length 5280 ))
    (/ D2* 12)))
  (setq B-v (round B-v ))
  (format t"
The total compacted Volume of Base material required  = ~d Cu-ft
  " B-v )
  (terpri)(terpri)
  (setf (get 'volume 'Base) Base-volume )
  (db-insert '(volume SubBase layer))
  (db-delete '(volume Base !j)))

```

```

;;                                     SubBase-volume
;;                                     =====
;; This Rule fires at its Priority(110), if the pattern (volume SubBase !j) matches.
;; It calculates the total compacted Volume of SubBase material required by using
;; required number of lanes, width of each lane, road length and D3*. It inserts pattern
;; to fire the rule "AC-Weight" and deletes its own pattern.

```

```

(rule 110 SubBase-volume (volume SubBase !j) -->
  (terpri)(terpri)
  (setq S.Base-v (round (* (* (* lanes lane-width) (* length 5280 ))
    (/ D3* 12))))
  (format t"
The total compacted Volume of SubBase material required = ~d Cu-ft
  " S.Base-v )
  (terpri)(terpri)
  (setf (get 'volume 'SubBase) S.Base-v)
  (db-insert '(Weight AC material))
  (db-delete '(volume SubBase !j)))

```

### **AC-Weight**

```
;;
;;
;; This Rule fires at its Priority(100), if the pattern (Weight AC !j) matches.
;; It calculates the total Weight of Asphalt Concrete Mix required by using
;; volume of Asphalt Concrete Mix and compacted density of Asphalt Concrete
;; layer. It inserts pattern to fire the rule "Base-Weight" and deletes its own pattern.
```

```
(rule 100 AC-Weight (Weight AC !j) -->
  (terpri)(terpri)
  (setq AC-wt (round (/ (* AC-v AC-density) 2240 )))
  (format t"
The total Weight of Asphalt Concrete Mix required = ~d Tons
" AC-wt)
  (terpri)(terpri)
  (setf (get 'Weight 'AC) AC-wt)
  (db-insert '(Weight Base layer))
  (db-delete '(Weight AC !j)))
```

### **Base-Weight**

```
;;
;;
;; This Rule fires at its Priority(90), if the pattern (Weight Base !j) matches.
;; It calculates the total Weight of Base material required by using volume of Base
;; material and compacted density of Base layer. It inserts pattern to fire the rule
;; "SubBase-Weight" and deletes its own pattern.
```

```
(rule 90 Base-Weight (Weight Base !j) -->
  (terpri)(terpri)
  (setq Base-wt (round (/ (* B-v Base-density) 2240 )))
  (format t"
The total Weight of Base material required      = ~d Tons
" Base-wt)
  (terpri)(terpri)
  (setf (get 'Weight 'Base) Base-wt)
  (db-insert '(Weight SubBase layer))
  (db-delete '(Weight Base !j)))
```

### **SubBase-Weight**

```
;;
;;
;; This Rule fires at its Priority(80), if the pattern (Weight SubBase !j) matches.
;; It calculates the total Weight of SubBase material required by using volume of
;; SubBase material and compacted density of SubBase layer. It inserts pattern to
;; fire the rule "cost-AC-layer" and deletes its own pattern.
```

```
(rule 80 SubBase-Weight (Weight SubBase !j) -->
  (terpri)(terpri)
  (setq SBase-wt (round (/ (* S.Base-v SubBase-density) 2240
    )))
  (format t"
The total Weight of SubBase material required    = ~d Tons
  " SBase-wt)
  (terpri)(terpri)
  (setf (get 'Weight 'SubBase) SBase-wt)
  (db-insert '(cost AC layer))
  (db-delete '(Weight SubBase !j)))
```

### **cost-AC-layer**

```
;;
;;
;; This Rule fires at its Priority(70), if the pattern (cost AC !j) matches. It calculates the
;; total cost of AC layer material by using Weight of Asphalt Concrete Mix and cost/ton
;; of AC layer material. It inserts pattern to fire the rule "cost-Base-layer" and deletes
;; its own pattern.
```

```
(rule 70 cost-AC-layer (cost AC !j) -->
  (terpri)(terpri)
  (setq AC-cost (round (* AC-wt AC-cost) ))
  (format t"
The cost of AC layer material    = ~d Dollars    " AC-cost)
  (terpri)(terpri)
  (setf (get 'AC 'cost) AC-cost)
  (db-insert '(cost base layer))
  (db-delete '(cost AC !j)))
```

```

;;                                     cost-Base-layer
;;                                     =====
;; This Rule fires at its Priority(60), if the pattern (cost Base !j) matches. It calculates the
;; total cost of Base layer material by using Weight of Base layer material and cost/ton
;; of base layer material. It inserts pattern to fire the rule "cost-SubBase-layer" and
;; deletes its own pattern.

```

```

(rule 60 cost-Base-layer (cost base !j) -->
  (terpri)(terpri)
  (setq base-cost (round (* base-wt base-cost) ))
  (format t"
The cost of Base layer material      = ~d Dollars      " base-cost)
  (terpri)(terpri)
  (setf (get 'base 'cost) base-cost)
  (db-insert '(cost Subbase layer))
  (db-delete '(cost base !j)))

```

```

;;                                     cost-SubBase-layer
;;                                     =====
;; This Rule fires at its Priority(50), if the pattern (cost SubBase !j) matches. It calculates
;; the total cost of SubBase layer material by using weight and cost/ton of Subbase layer
;; material. It inserts pattern to fire the rule "Lane-mile-cost" and deletes its own pattern.

```

```

(rule 50 cost-SubBase-layer (cost Subbase !j) -->
  (terpri)(terpri)
  (setq Subbase-cost (round (* Sbase-wt Subbase-cost) ))
  (format t"
The cost of SubBase layer material   = ~d Dollars   " Subbase-cost)
  (terpri)(terpri)
  (setf (get 'Subbase 'cost) Subbase-cost)
  (db-insert '(lane cost value))
  (db-delete '(cost Subbase !j)))

```





```

;;                                     Lane-mile-cost
;;                                     =====
;; This Rule fires at its Priority(40), if the pattern (lane cost !j) matches. It calculates the
;; cost of one mile of each lane by using total cost of AC layer material, Base and
;; Subbase layer material, number of lanes and road length. It inserts pattern to fire
;; the rule " Show-results " and deletes its own pattern.

```

```

(rule 40 Lane-mile-cost (lane cost !j) -->
  (terpri)(terpri)
  (setq lanemile-cost (round (+ (/ (+ AC-cost base-cost Subbase-cost)
    lanes) length) Overhead-plm )))
  (format t"
The cost of one mile of each lane , having ~d inches thick AC layer ,

~d inches thick Base layer, and ~d inches thick SubBase layer (including

~d Dollars/lane-mile Overhead charges)    = ~d Dollars/lane-mile  "
  D1* D2* D3* Overhead-plm lanemile-cost)
  (terpri)(terpri)
  (setf (get 'lane 'cost) lanemile-cost)
  (db-insert '(Show design results))
  (db-delete '(lane cost !j)))

```

```

;;                                     Show-results
;;                                     =====
;; This Rule fires at its Priority(30), if the pattern (Show design !j) matches. It shows the
;; results of pavement design in graphical form by giving the SN values and layer
;; thickness also. It inserts pattern to fire the rule "total-cost" and deletes its own pattern.

```

```

(rule 30 Show-results (Show design !j) -->
  (terpri)(terpri)
  (format t"

```

## THICKNESS OF LAYERS

Surface, (inches)	= ~d
Base, (inches)	= ~d
SN1	= ~d
Sub-base, (inches)	= ~d
SN2	= ~d
Sub-grade	
SN3	= ~d
////////////////////	

"

D1\* D2\* SN1 D3\* SN2 SN3 )  
 (db-insert '(total cost road))  
 (db-delete '(Show design !j)))

```

;;                                     total-cost
;;                                     =====
;; This Rule fires at its Priority(20), if the pattern (total cost !j) matches. It calculates the
;; total cost of Road by using total cost of AC layer material, Base and Subbase layer
;; material and Overhead charges. It ends the program.

```

```

(rule 20 total-cost (total cost !j) -->
  (terpri)(terpri)
  (setq T-cost (round (+ AC-cost base-cost Subbase-cost Total-Overhead)))
  (format t"
    The Total cost of ~d miles long road having ~d lanes (each ~d feet wide)

                                = ~d Dollars    "
    length lanes lane-width T-cost)
  (format t"

----- END OF RESULTS -----  ")
  (terpri)(terpri)
  (db-delete '(total cost !j)))

```

**Appendix-D****ROAD DESIGNING - EXAMPLE 1**

<4 turmeric:~ >*emacs -1.5.2*

---

---

Starting image `/opt/allegro-cl/bin/allegro-cl'  
 with no arguments  
 in directory `/user/bashirmu/lisp/'  
 on machine `pacific'.

Allegro CL 4.2 [SPARC; R1] (10/10/94 11:02)  
 Copyright (C) 1985-1993, Franz Inc., Berkeley, CA, USA. All Rights Reserved.  
 ;; Optimization settings: safety 1, space 1, speed 1, debug 2  
 ;; For a complete description of all compiler switches given the current  
 ;; optimization settings evaluate (EXPLAIN-COMPILER-SETTINGS).  
  
 ;; Starting socket daemon and emacs-lisp interface...

USER(1): (*load "bashroad"*)  
 ; Loading /user/bashirmu/lisp/bashroad.  
 T

USER(2): (*bashroad*)

WELCOME TO 'BASHROAD' FOR

---

---

DESIGNING OF ROADS

---

---

THIS PROGRAMME WILL GUIDE YOU TO ESTIMATE THE THICKNESS OF DIFFERENT LAYERS OF FLEXIBLE PAVEMENT/ SPECIFICATIONS OF A REQUIRED ROAD KEEPING IN VIEW YOUR REQUIREMENTS. IT CAN ALSO GIVE THE ESTIMATED QUANTITIES/COST OF DIFFERENT MATERIALS AND THE TOTAL COST OF THE ROAD (INCLUDING PER LANE-MILE COST).

" PLEASE ENTER THE TOTAL NUMBER OF LANES REQUIRED FOR THE ROAD  
(WRITE INTEGER ONLY).(EXAMPLE VALUE IS ' 4 ' ) "

**4**

" PLEASE ENTER WIDTH OF EACH LANE IN FEET. (WRITE INTEGER ONLY).  
(TYPICAL VALUE IS ' 12 ' FEET) "

**12**

" PLEASE ENTER THE LENGTH OF THE REQUIRED ROAD IN MILES  
(WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 10 ' ) "

**10**

PLEASE ENTER A NUMBER FROM 1 TO 3.

1. TO ENTER THE STRUCTURAL NUMBER REQUIRED TO PROTECT  
DIFFERENT LAYERS.
2. BashRoad SHOULD CALCULATE THE STRUCTURAL NUMBER  
REQUIRED.
3. TO QUIT BASH-ROAD.

**2**

PLEASE ENTER THE DESIGN PERIOD IN YEARS (WRITE NUMBER ONLY).  
EXAMPLE VALUE IS ' 10 ' YEARS.

**10**

PLEASE ENTER THE TWO WAY TRAFFIC I.E EQUIVALENT SINGLE AXEL  
LOAD (ESAL) FOR 10 YEARS IN MILLIONS. EXAMPLE VALUE FOR 10 YEARS  
IS ' 32 ' MILLIONS. IF YOU WANT TO GIVE INITIAL YEAR TRAFFIC,  
ENTER ' I ' .

**i**

PLEASE ENTER THE TWO WAY TRAFFIC (ESAL) FOR FIRST YEAR  
IN MILLIONS. EXAMPLE VALUE IS ' 2.67 ' MILLIONS.

**2.67**

PLEASE ENTER THE ANNUAL GROWTH RATE OF TRAFFIC IN PERCENT .  
EXAMPLE VALUE IS ' 4 ' PERCENT .

**4**

PLEASE ENTER THE DIRECTIONAL DISTRIBUTION FACTOR. TYPICAL  
VALUE FOR TWO WAY TRAFFIC IS ' 0.5 ' .

**0.5**

PLEASE ENTER THE LANE DISTRIBUTION FACTOR FOR OUTER/DESIGN  
LANE. EXAMPLE VALUE IS ' 0.70 ' .

**0.7**

PLEASE ENTER THE OVERALL STANDARD DEVIATION. EXAMPLE VALUE  
IS ' 0.45 ' .

**0.45**

PLEASE ENTER THE DESIGN RELIABILITY IN PERCENT. TYPICAL VALUE IS  
' 95 ' PERCENT .

**30**

DESIGN RELIABILITY SHOULD BE BETWEEN 50 AND 99.99 PERCENT .  
PLEASE ENTER THE DESIGN RELIABILITY IN PERCENT. TYPICAL VALUE IS  
' 95 ' PERCENT .

**95**

PLEASE ENTER THE OVERALL SERVICEABILITY LOSS . EXAMPLE VALUE  
IS ' 2 ' . IF YOU WANT TO GIVE INITIAL AND TERMINAL PAVEMENT  
SERVICEABILITY INDEX, ENTER ' I ' .

**i**

PLEASE ENTER THE INITIAL PAVEMENT SERVICEABILITY INDEX.  
EXAMPLE VALUE IS ' 4.5 ' .

**6**

INITIAL PAVEMENT SERVICEABILITY INDEX ABOVE 4.7 IS IMPROBABLE .  
PLEASE ENTER THE INITIAL PAVEMENT SERVICEABILITY INDEX.  
EXAMPLE VALUE IS ' 4.5 ' .

**0.5**

INITIAL PAVEMENT SERVICEABILITY INDEX BELOW 1.5 IS NOT  
ADVISABLE AS THE ROAD WILL NOT BE SUITABLE FOR TRAFFIC.

PLEASE ENTER THE INITIAL PAVEMENT SERVICEABILITY INDEX.  
EXAMPLE VALUE IS ' 4.5 ' .

**4.5**

PLEASE ENTER THE TERMINAL PAVEMENT SERVICEABILITY INDEX.  
EXAMPLE VALUE IS ' 2.5 ' .

**2.5**

PLEASE ENTER THE SERVICEABILITY LOSS DUE TO ENVIRONMENT.  
EXAMPLE VALUE IS ' 0.64 ' . ENTER ' 0 '(ZERO) TO IGNORE THE  
ENVIRONMENTAL EFFECTS .

**0.64**

"PLEASE ENTER THE ASPHALT CONCRETE MODULUS OF ELASTISITY IN PSI.  
(WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 300,000 ' "

**MR=300000**

"YOU HAVE NOT ENTERED A NUMBER"

"PLEASE ENTER THE ASPHALT CONCRETE MODULUS OF ELASTISITY IN PSI.  
(WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 300,000 ' "

**300000**

"PLEASE ENTER THE GRANULAR BASE RESILIENT MODULUS IN PSI. (WRITE  
NUMBER ONLY).(EXAMPLE VALUE IS ' 25,000 ' "

**25000**



"PLEASE ENTER THE GRANULAR SUBBASE RESILIENT MODULUS IN PSI.  
(WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 12,000 ' "  
**12000**

"PLEASE ENTER THE ROAD BED SOIL RESILIENT MODULUS IN PSI. (WRITE  
NUMBER ONLY).(EXAMPLE VALUE IS ' 2,200 ' "  
**2200**

"PLEASE ENTER THE DRAINAGE MODIFYING FACTOR FOR BASE LAYER .  
EXAMPLE VALUE IS ' 1.2 ' "  
**1.2**

"PLEASE ENTER THE DRAINAGE MODIFYING FACTOR FOR SUB-BASE  
LAYER . EXAMPLE VALUE IS ' 0.7 ' "  
**0.7**

" DO YOU WANT THE ESTIMATED QUANTITY AND COST OF MATERIL ALSO  
(IN  
ADDITION TO LAYER THICKNESSES ) 'Y' or 'N' "  
**n**

---

---

## DESIGN RESULTS

---

---

Structural Number to protect the Base layer =  $SN1 = 3.6999986$

Structural Number to protect the Sub-Base layer =  $SN2 = 4.8999977$

Structural Number to protect the Road Bed Soil =  $SN3 = 8.299995$

" The value of layer coefficient for AC =  $a1 = 0.3754528$

" The value of layer coefficient for BASE =  $a2 = 0.11808705$

" The value of layer coefficient for SUB-BASE =  $a3 = 0.086974144$

Calculated thickness of the AC layer =  $D1 = 9.854764$  inches

Rounded-Up thickness of the AC layer =  $D1^* = 10.0$  inches

The actual Structural Number for AC layer =  $SN1^* = 3.7545278 \geq SN1(3.6999986)$   
 $\Rightarrow$  OK

Calculated thickness of the Base layer =  $D2 = 8.083513$  inches

Rounded thickness of the Base layer =  $D2^* = 8$  inches

The actual combined Structural Number of the Surface layer and Base course is not greater than the Structural Number of the Sub-Base course

i.e  $(SN1^* + SN2^*) < SN2$  Or Numerically ,

$$(3.7545278 + 1.1336358)(= 4.8881636) < 4.8999977 \implies \text{Incorrect}$$

So increasing the thickness of the Base layer by 1 inch

Now increased thickness of the Base layer =  $D2^* = 9$  inches

The actual combined Structural Number of the Surface layer and Base course is greater than the Structural Number of the Sub-Base course i.e

$(SN1^* + SN2^*) \geq SN2$  Or Numerically ,

$$(3.7545278 + 1.2753402)(= 5.029868) \geq 4.8999977 \implies \text{OK}$$

Calculated thickness of the SubBase layer =  $D3 = 53.712635$  inches

Rounded thickness of the SubBase layer =  $D3^* = 54$  inches

The actual combined Structural Number of the Surface layer and Base and SubBase is greater than the Structural Number of the Sub-Base course

i.e  $(SN1^* + SN2^* + SN3^*) \geq SN3$  Or Numerically ,

$$(3.7545278 + 1.2753402 + 3.2876227)(= 8.317491) \geq 8.299995 \implies \text{OK}$$

**Appendix-E****ROAD DESIGNING - EXAMPLE 2**

[1] USER(3): (*load "bashroad"*)  
 ; Loading /user/bashirmu/lisp/bashroad.  
 T

[1] USER(4): (*bashroad*)

WELCOME TO 'BASHROAD' FOR  
 =====  
 DESIGNING OF ROADS  
 =====

THIS PROGRAMME WILL GUIDE YOU TO ESTIMATE THE THICKNESS OF DIFFERENT LAYERS OF FLEXIBLE PAVEMENT/ SPECIFICATIONS OF A REQUIRED ROAD KEEPING IN VIEW YOUR REQUIREMENTS. IT CAN ALSO GIVE THE ESTIMATED QUANTITIES/COST OF DIFFERENT MATERIALS AND THE TOTAL COST OF THE ROAD (INCLUDING PER LANE-MILE COST).

" PLEASE ENTER THE TOTAL NUMBER OF LANES REQUIRED FOR THE ROAD  
 (WRITE INTEGER ONLY).(EXAMPLE VALUE IS ' 4 ' ) "

**4**

" PLEASE ENTER WIDTH OF EACH LANE IN FEET. (WRITE INTEGER ONLY).  
 (TYPICAL VALUE IS ' 12 ' FEET) "

**12**

" PLEASE ENTER THE LENGTH OF THE REQUIRED ROAD IN MILES  
 (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 10 ' ) "

**10**

PLEASE ENTER A NUMBER FROM 1 TO 3.

1. TO ENTER THE STRUCTURAL NUMBER REQUIRED TO PROTECT DIFFERENT LAYERS.
2. BashRoad SHOULD CALCULATE THE STRUCTURAL NUMBER REQUIRED.
3. TO QUIT BASH-ROAD.

**2**

PLEASE ENTER THE DESIGN PERIOD IN YEARS (WRITE NUMBER ONLY).  
EXAMPLE VALUE IS ' 10 ' YEARS.

**10**

PLEASE ENTER THE TWO WAY TRAFFIC I.E EQUIVALENT SINGLE AXEL LOAD(ESAL) FOR 10 YEARS IN MILLIONS. EXAMPLE VALUE FOR 10 YEARS IS ' 32 ' MILLIONS. IF YOU WANT TO GIVE INITIAL YEAR TRAFFIC, ENTER ' I ' .

**32**

PLEASE ENTER THE DIRECTIONAL DISTRIBUTION FACTOR. TYPICAL VALUE FOR TWO WAY TRAFFIC IS ' 0.5 ' .

**0.5**

PLEASE ENTER THE LANE DISTRIBUTION FACTOR FOR OUTER/DESIGN LANE. EXAMPLE VALUE IS ' 0.70 ' .

**0.7**

PLEASE ENTER THE OVERALL STANDARD DEVIATION. EXAMPLE VALUE IS ' 0.45 ' .

**0.45**

PLEASE ENTER THE DESIGN RELIABILITY IN PERCENT. TYPICAL VALUE IS ' 95 ' PERCENT .

**95**

PLEASE ENTER THE OVERALL SERVICEABILITY LOSS . EXAMPLE VALUE IS ' 2 ' . IF YOU WANT TO GIVE INITIAL AND TERMINAL PAVEMENT SERVICEABILITY INDEX, ENTER ' I ' .

**2**

PLEASE ENTER THE SERVICEABILITY LOSS DUE TO ENVIRONMENT. EXAMPLE VALUE IS ' 0.64 ' . ENTER ' 0 '(ZERO) TO IGNORE THE ENVIRONMENTAL EFFECTS .

**0.64**

"PLEASE ENTER THE ASPHALT CONCRETE MODULUS OF ELASTISITY IN PSI. (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 300,000 ' "

**300000**

"PLEASE ENTER THE GRANULAR BASE RESILIENT MODULUS IN PSI. (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 25,000 ' "

**25000**

"PLEASE ENTER THE GRANULAR SUBBASE RESILIENT MODULUS IN PSI. (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 12,000 ' "

**12000**

"PLEASE ENTER THE ROAD BED SOIL RESILIENT MODULUS IN PSI. (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 2,200 ' "

**2200**

"PLEASE ENTER THE DRAINAGE MODIFYING FACTOR FOR BASE LAYER . EXAMPLE VALUE IS ' 1.2 ' "

**1.2**

"PLEASE ENTER THE DRAINAGE MODIFYING FACTOR FOR SUB-BASE LAYER . EXAMPLE VALUE IS ' 0.7 ' "

**0.7**

" DO YOU WANT THE ESTIMATED QUANTITY AND COST OF MATERIL ALSO (IN ADDITION TO LAYER THICKNESSES ) 'Y' or 'N' "

**y**

"PLEASE ENTER THE COMPACTED DENSITY OF ASPHALT CONCRETE LAYER (IN Lbs/cu-ft)(WRITE NUMBER ONLY).(TYPICAL VALUE IS ' 150 ' Lbs/cu-ft) "

**150**

"PLEASE ENTER THE COMPACTED DENSITY OF BASE LAYER (IN Lbs/cu-ft) (WRITE NUMBER ONLY).(TYPICAL VALUE IS ' 135 ' Lbs/cu-ft) "

**135**

"PLEASE ENTER THE COMPACTED DENSITY OF SUB-BASE LAYER (IN Lbs/cu-ft) (WRITE NUMBER ONLY).(TYPICAL VALUE IS ' 125 ' Lbs/cu-ft) "

**125**

PLEASE ENTER THE COST OF ASPHALT CONCRETE MATERIAL IN DOLLARS/TON (WRITE NUMBER ONLY).(TYPICAL RANGE IS BETWEEN ' 15 ' AND ' 50 ' DOLLARS/TON)

**30**

PLEASE ENTER THE COST OF BASE MATERIAL IN DOLLARS/TON (WRITE NUMBER ONLY).(TYPICAL RANGE IS BETWEEN ' 8 ' AND ' 25 ' DOLLARS/TON)

**15**

PLEASE ENTER THE COST OF SUB-BASE MATERIAL IN DOLLARS/TON  
(WRITE NUMBER ONLY).(TYPICAL RANGE IS BETWEEN ' 3 ' AND ' 12 '  
DOLLARS/TON)

**8**

"PLEASE ENTER THE OVERHEAD CHARGES FOR 1 MILE OF EACH LANE(IN  
DOLLARS /LANE/MILE) (WRITE NUMBER ONLY).(EXAMPLE VALUE IS ' 7000 '  
DOLLARS /LANE-MILE) "

**7000**

---

---

### DESIGN RESULTS

---

---

Structural Number to protect the Base layer = SN1 = 3.6999986

Structural Number to protect the Sub-Base layer = SN2 = 4.8999977

Structural Number to protect the Road Bed Soil = SN3 = 8.299995

" The value of layer coefficient for AC =a1 = " 0.3754528



" The value of layer coefficient for BASE =  $a_2 = 0.11808705$

" The value of layer coefficient for SUB-BASE =  $a_3 = 0.086974144$

Calculated thickness of the AC layer =  $D_1 = 9.854764$  inches

Rounded-Up thickness of the AC layer =  $D_1^* = 10.0$  inches

The actual Structural Number for AC layer =  $SN_1^* = 3.7545278 \geq SN_1(3.6999986)$   
 $\Rightarrow$  OK

Calculated thickness of the Base layer =  $D_2 = 8.083513$  inches

Rounded thickness of the Base layer =  $D_2^* = 8$  inches

The actual combined Structural Number of the Surface layer and Base course is not greater than the Structural Number of the Sub-Base course

i.e.  $(SN_1^* + SN_2^*) < SN_2$  Or Numerically ,

$$(3.7545278 + 1.1336358) (= 4.8881636) < 4.8999977 \Rightarrow \text{Incorrect}$$

So increasing the thickness of the Base layer by 1 inch

Now increased thickness of the Base layer =  $D_2^* = 9$  inches

The actual combined Structural Number of the Surface layer and Base course is greater than the Structural Number of the Sub-Base course i.e

$$(SN1^* + SN2^*) \geq SN2 \quad \text{Or Numerically ,}$$

$$(3.7545278 + 1.2753402)(= 5.029868) \geq 4.8999977 \implies \text{OK}$$

Calculated thickness of the SubBase layer =  $D3 = 53.712635$  inches

Rounded thickness of the SubBase layer =  $D3^* = 54$  inches

The actual combined Structural Number of the Surface layer and Base and SubBase is greater than the Structural Number of the Sub-Base course

$$\text{i.e } (SN1^* + SN2^* + SN3^*) \geq SN3 \quad \text{Or Numerically ,}$$

$$(3.7545278 + 1.2753402 + 3.2876227)(= 8.317491) \geq 8.299995 \implies \text{OK}$$

The total compacted Volume of Asphalt Concrete Mix required = 2112000.0 Cu-ft

The total compacted Volume of Base material required = 1900800 Cu-ft

The total compacted Volume of SubBase material required = 11404800 Cu-ft

The total Weight of Asphalt Concrete Mix required = 141429 Tons

**The total Weight of Base material required = 114557 Tons**

**The total Weight of SubBase material required = 636429 Tons**

**The cost of AC layer material = 4242870 Dollars**

**The cost of Base layer material = 1718355 Dollars**

**The cost of SubBase layer material = 5091432 Dollars**

**The cost of one mile of each lane , having 10.0 inches thick AC layer ,  
9 inches thick Base layer, and 54 inches thick SubBase layer (including  
7000 Dollars/lane-mile Overhead charges) = 283316 Dollars/lane-mile**

**THICKNESS OF LAYERS**


---

Surface, (inches) = 10.0

---

Base, (inches) = 9

SN1 = 3.6999986

---

Sub-base, (inches) = 54

SN2 = 4.8999977

---

Sub-grade

SN3 = 8.299995

---

////////////////////////////////////

The Total cost of 10 miles long road having 4 lanes (each 12 feet wide)

= 11332657 Dollars

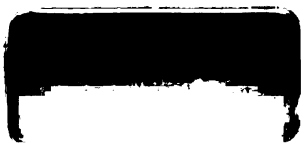
----- END OF RESULTS -----

NIL

[1] USER(5):

## **LIST OF REFERENCES**

1. [YODER, E.J., and WITCZAK, M.W. : ‘ Principles of pavement design ‘ (John Wiley, New York, 1975) pp. 504-519].
2. [ AASHTO. : ‘Interim guide for design of pavement structures’. American Association of State Highway and Transportation, Washington, 1981].
3. [DARTER, I., MICHAEL, W., HUDSON, R. : ‘ Selection of optimal pavement designs considering reliability, performance. and cost ‘ , Trans. Res., 1974, 485, ]
4. [American Association of State, Highway, and Transportation Officials.  
DNPS86. A Computer Program to Perform the AASHTO Design Procedure,  
1986]
5. Peter Norvig, “Paradigms of Artificial Intelligence Programming: Case studies in Common Lisp, ” 1992.
6. Robert Wilensky, “ Common LISPcraft , ” University of California, Berkeley.  
W.W. Norton & Company New York London., 1986.



MICHIGAN STATE UNIV. LIBRARIES



31293013893668