(1636)

AUTOBEAM

AN EXPERT SYSTEM TO ANALYSE AND DESIGN R.C.C. BEAMS

Dissertation submitted to the Jawaharlal Nehru University
in partial fulfilment of the requirements for
the award of the Degree of
MASTER OF TECHNOLOGY

V.P

JAGADESH . P . NANISETTY

Sup : Balasundaram, S

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI

February 1988

TO

P R O L O G

C E R T I F I C A T E

This work, embodied in the dissertation titled,

AUTO BEAM

AN EXPERT SYSTEM TO ANALYSE AND DESIGN R.C.C. BEAMS

has been carried out by Mr. Jagadesh.P.Narisetty bonafied student of School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi.

This work is original and has not been submitted for any degree or diploma in any other University or Institute.

29/2/88

Dr. S. Balasundaram
Asst. Professor
School of Computer and
Systems Sciences
Jawaharlal Nehru University
New Delhi.

Prof. Karmeshu
Dean
School of computer and Systems Sciences
Jawaharlal Nehru University
New Delhi

ACKNOWLEDGEMENT

I am very much indebted to my guide, **Dr. S. Balasundaram,** Asst. Professor, who has been extremely helpful and encouraging through out the project, without which it would have been very difficult to complete the project.

Dr. Ashok Gupta, Asst. Professor, Indian Institute of Technology, New Delhi, has played a very significant role by giving his timely and extremely useful suggestions and sparing his precious time disscussing with us.

I thank **Mr. D. Jagadesh, Mr. Seshu, Mr. Pinakapani** for their help and advices.

I also thank **Mr. Fateh Singh,** Administrative officer of our school for his cooperation.

I thank our dean **Prof. Karmeshu** , who is very encouraging through out this project.

Jagadesh. P. Nanisetty

SYNOPSIS

The quest to automate the civil engineering design process has lead the designer to opt for computers. Civil engineering design problems characterised by their inherent impression, paucity incompleteness of data, and heavy relience on expert views, defys the algorithmic approach, since the experience and intutive judgement form an important role in the design process. So the need of a software which posseses the knowledge of an expert is inevitable, which guides the designer towards the solution.

The Expert systems are the computer programs which were built into the knowledge and are capable to operate at the expert level. The Expert System naturally contains large and varied knowledge about a specific area, from which the inference mechanism infers the context based inferences.

PROLOG has a inbuilt backward chaining inference mechanism, capable of representing the fuzzy and imprecise knowledge and caple of exploring the parallel processing archilecures, promotes it to develop Expert Systems.

The problem of designing the reinforced concrete beams demands rigorous numerical computation to analyse the beam and a knowledge base containing the knowledge and the rules, formulae and specifications of design, and a inference mechanism which infers context based conclusions.

It is not easy to implement both the symbolic and numerical computations using a single programing language. PROLOG is used to implement the symbolic computations and PASCAL used to implement the numerical computations. The linking and data transfer is through I/O files. The nemerical computations routine implemented in PASCAL is executed in PROLOG environment.

This program posseses all the knowledge needed to analyse and design any of the cantilever, simply supported, over hangining and fixed beams with any number of point or uniformly distributed loads.

Analysis part of the program implemented in PASCAL analyses all the loads and displays the output graphically and transfers the data into a file. This data file is read in PROLOG environment and the data is stored in the knowledge base as context based facts. The design, implemented in PROLOG is based on the IS-456 codal specifications. The section of the beam and the reinforcement details which are satisfactory to the user and not contradicting the code is arrived at. All the drawing are displayed graphically.

CONTENTS

CHAPTER.1. THE NEED OF EXPERT SYSTEM

The present civil engineer is confronted with new challenges in the design process due to the increasing interest of the pubic towards a complicated geometry and architectures, which are still encouraged by the improving construction techniques and materials of use. The quest to automate the design process had lead the designer to opt for computers to cope with the limited time provided for the design. Algorithms were developed to design problems, but here are aspects of design however which seems to defy algorithmic approach.

One technique that has been around for design and is seriously considered, is optimization technique. Considering the geometry of the structre .ie. beams, columns, slabs etc. a penalty functon which, if minimized the optimized solution is arrived at, is identified. Generally in civil engineering problems cost is the penalty function, but can be the construction time also.

The constraints which the structre must satisfy viz. limiting stress, deflection and moment of resistance etc. which are governed by the code of practice and also the structure design feasibility of the complexity are identified.

There can be many solutions which satisfy the constraints, out of which one has to be opted, this obviously requires experience and precise knowledge of the area, rigorous numerical computations for each design and many such designs should be prepared and finally to get optimum solution satisfying all the constraints.

In the inital stage of design ,vialble alternatives are compared and one selected. An inappropriate solution often leads to severe consequences. It is quite natural that the problem is ill defined since the requirements are often finite and the vague factors are to be fixed, depending on the design fulfilling the constraints.

It needs a vast amount of experience and heuristic knowledge for the design. Analytical solutions given by the conventional software do not help much, since a conceptual procedure is desired.

A quite natural problem the designer faces in the design process is the discontinuous constranits, if the constraints are discontinuous and liable to change with each design will obviously need an experienced heuristic knowledge. It is better to have a knowledge base which contains the knowledge of the specified field and a inference mechanism which can act upon the KB and pick up the relevent rules and infer the best possible conclusions.

The experience and heuristic knowledge is still indispensible and not possesed by all. Thus the need of software which contains the knowledge of an experienced engineer is evitable. If not, a fulfledged but a software which can help, advice and guide the designer to better solution and design.

CHAPTER.3. INTRODUCTION TO EXPERTS SYSTEMS

ES is a computer program that has built into the knowledge and capability that will allow it to operate at the experts level. ES naturally contain large amount of varied knowledge and rules of the specific area and not only the rules and knowledge also but the heuristic knowledge which is the knowledge of the practical experience.

Expert system operate particularly well where the thinking is mostly reasoning not calculating, and that means most of the world knowledge. Even though a lot of professional work seems to be expressed in mathematical formulae, in fact, except in mathematically based sciences, the difficult choices, the matters that set experts apart from beginners, are symbolic and inferential, which are rooted in experimential knowledge. Human experts have acquired their expertise not only from explicit knowledge found in textbooks and lectures, but also from experience by doing things again and again, failing, succeeding, wasting time and effort, then learning to save them, getting a feel for a problem. Learning when to go by the book and when to break the rules. They therefore build up a teritory of working rules of thumb, or "heuristics" that, combined with book knowoledge, make them expert practitioners.

Perhaps the largest single group of expert systems is centered in medicine. The most knowledge-intensive expert system in existence is the INTERNIST CADUCEUS system at the University of Pittsburgh, the creation of a physician, Jack Meyers, and a computer scientist, Harry

Pople. INTERNIST CADUCEUS does diagnoses in internal medicine at a level of expertise that allows it to solve most of the CPCs, or clinical pathological conferences, INTERNIST covers more than 80 percent of all internal medicine; its knowledge base encompasses about 500 diseases and more than 3,500 manifestations of disease.

Although INTERNIST CADUCEUS was designed to aid skilled internists in complicated medical problems, the program will probably have a future life as a diagnostic aid to physician assistantsd and in rural health clinics, in military medicine, and in space travel.

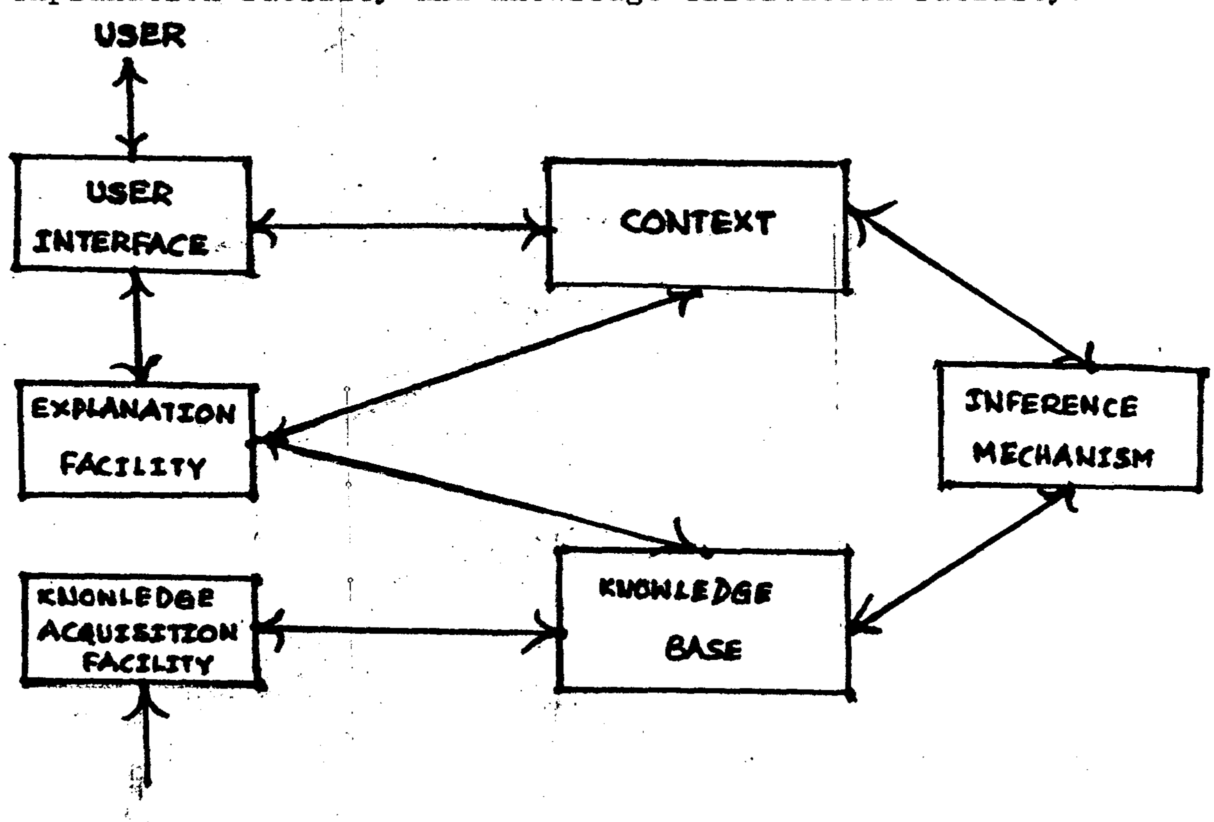
At Stanford University, several medical expert systems have been designed. MYCIN diagnoses blood and maningitis infections, then advises the physician on antibiotic therapies for treating the infections. Like every other expert system. MYCIN acts as a consultant, having a conversation with its user. the physicia. The physician supplies the patient history and laboratory test results-external data the computer couldn't possibly infer-and then the program begins to reason about possible diagnoses. If the physician is uncertain why the program has arrived at a given diagnosis, or why certain drugs have been suggested as therapy, he can ask the program for its line of reasoning.

There is no one specilist who expert ise spans the whole problem. It can only be solved by the interaction of several specilist and the interaction of several specialists and the intelligent fusion of they seperate expertise. Not always but sometime expert system can manage the intrinsic it is complexity of problems better than human expert. It is this is specifically true of problems that are combinational involving a great deal of trial an error. Trying out combination of problems elements systematically. Problems of design and configuaration examples as are problems of data analysis hypothesis and diagnosis.

DEC engineers use a configuration expert system to plan and manufacture their VAX computers. The systems is reported to plan corruptly in more than 99 percent of the gases. "Prospector" expert system is used by the geologist advising them in exploration of minerals.

ANOTOMY OF EXPERT SYSTEM

An expert system can contain knowledge-base, inference mechanism, contest, user, interface, and explanation facility and knowledge elicitation facility.



KNOWLEDGE-BASE: The knowledge base contains knowledge specific to the domain of the problem. The knowledge is of two types :

(1) The facts of the domain, a widely shared knowledge, commonly agreed among practitioners that is written in test books and generals of the field.

(2) The heuristic knowledge, which is a knowledge of good practice and good judgement in a field it is basically knowledge of experience which is acquired over years of practice. This heuristic knowledge gives good guessing which avoids unnecessary such and calculation and leads to the solution. The knowledge base should contain both the knowledge specified above. Expert system containing facts doesn't mean a fulfledged expert system because expert system is supposed to contain the knowledge of the expert, expert becomes an expert if he can judge or guess correctly without trying all possible solutions and picking up the optimum one.

In addition to knowledge, an expert system needs an inference procedure a method of of reasoning used to understand and act upon the combination of knowledge and problem data. The inference procedures, or problem-solving methods, used by knowledge engineers do not need to be arcane or complex,. Even simple methods, used in commonsense reasoning or taught in first course in logic, are adequate. In fact, there is a virtue in employing simple inference procedures since they are easily understood

Now we are able to be more precise about the problem of machine learning, and with this increased precision has come a new term, knowledge acquisition research.

This is the most important of the central problems of artificial intelligence research. The reason is simple: the power to enhance or amplify the programance of AI programs resides in the specific knowledge of the problem domain that can be brought to bear. Thus, efficient knowledge bases must be large and of high quality.

This knowledge is currently acquired in a very painstaking way, individual computer scientists work with individual experts to explicate the experts heuristics domins those jewels of knowledge out of their heads one by one. If applied AI is to be important in the decades to come and we believe it must develop more automatic means for what is currently a very tedious, time consuming and expensive procedures. Right now, the problem of knowledge acquisition is the critical bottle neck in artificial intelligence.

updated in terms of its features and properties and their relationships with each other in a knowledge base? These and other tasks need to be done automatically within the system.

In sum, scientific issues central to artificial intelligence underline knowledge engineering and can be enumerated as the parts of an expert system. First it the problem of knowledge representation. How shall the knowledge of a domain of work be represented as data structures in the memory of the computer in a manner in which they can be conveniently accessed for problem solving?

Second is the problem of knowledge utilization. How can this knowledge be used in problem solving? In other words, how should the inference engine be designed?

Third, the most important, is the question of knowledge acquisition. How is it possible to acquire the knowledge so important for problem solving automatically, or at least semiautomatically, in a way in which the computer eases the transfer of expertise from human to the symbolic data structures that constitute the knowledge representation in the machine? Knowledge acquisition is a long-standing problem of AI. Like 'intelligence' learning has proven to be a catchall term that's too vague to be useful in creating intelligent computer programs. It served no better purpose to ask whether a machine really could be said to 'learn' than to ask whether a machine really could be said 'think' even when it improved its behaviour by experience (as one of the earliest programs isn artificial intelligence had done, a program that eventually played championship checkers).

far as possible. This principle is called the least commitment principle because variables are not instantiated until more information about the problem space is available.

Constraint Handling : If the subgoals of the hierarchial planning do not interact with each other, they can be solved independently. However, in practice these subgoals do interact. The interaction between subgoals can be handled by constraint satisfaction method. Constraint satisfaction methods involve the determination of problem states that satisfy given set of constraints. Essentially this method utilizes constraints to determine the values of parameter in a completely specified problem.

MANAGING THE KNOWLEDGE

For example, one simple form of reasoning that is commonly used is goal-directed backward chaining, the common mental strategy of "working backward" from a desired goal to what you know about achieving it at your starting point.

Al researchers have identified, dissected, and then replicated many such procedures that human beings use all the time, and knowledge engineers, who build expert systems, are skilled at choosing the right set of inference procedures for the type of program they are writing.

An expert system also requires methods of representing the knowledge it is to contain. This is a technical issue and a matter of some professional dispute, but essentially it means that both a logical structure and a set of appropriate data structured are necessary, through which the special knowledge in the knowledge base can find its way into the memory of a computer.

There is also a formidable problem of knowledge-base management, analogous to data-base management. How shall knowledge be organized, controlled propagated, as well as and used to find an operator most relevent to reducing this difference. If the operator is not directly applicable to the current situation, then the problem state is changed by setting up sub goals, so that the operator can be applied. After an operator has been applied the current state corresponds to a modified state. Means-ends analysis utilizes the forward and the backwarding techniques.

Problem reduction : Problem reduction involves factoring problems into smaller subproblems. The problem is represented by means of an AND-OR graph. An AND node consists of arcs pointing to a number of succesor nodes, all of which must be solved for the AND node to be true. For an OR node, it is sufficient for one of the succesor nodes to be solved, an OR node indicates that a number of alternative solutions exist for the problem. In many cases, backward chaining is used to solve the AND-OR graph.

Hierarchial Planning : The concept of hierarchial planning involves developing a plan at succesive levels of abstraction. In the design of complex systems, the design space is divided into a set of levels where the higher levels are abstractions of details at lower levels, the problem is hierarchially decomposed into loosely coupled subsystems. A number of solutions may exist for each sub system. However, enough information may not be available to assertain various variables of the subsystem. Further, the solution to one subproblem may depend on the decisions made in the solution of another subsystem. To minimize this dependency, it is important to defer binding decisions as by end-users, the people being assisted by expert systems. When these users are reviewing the system's line of reasoning. End-users will not come to trust the reasoning of an expert system, and therefore will not use it. Unless they can easily understand what it is doing.

PROBLEM SOLVING STRATAGIES

Problem solving involves the search for a solution through a state space by the application of operators, where the state space consists of an initial state, a goal state and intermediate states. The solution consists of all states that lead from the initial state to goal state.

Forward chaining: A system is said to exibit forward chaining also called as botton-up, data-driven, antecedent-driven, if it works from as initial state of known facts to a goal state. Here all facts ate input to the system and the system deduces the almost appropriate hypothesis or goal state that fits the facts. The main drawback of his strategy is that it is extremely wasteful to require as input data all the possible facts for all conditions in many circumstances all possible facts are not knowm or relevent. Some times the problem solving strategy mechanism is guided by the forwrd chaining is called event-driven. The forward chaining strategy is not appropriate for a design problem if possible goal states of the design problems are not easily represented by a discrete nember of hypothesis.

Backward chaining: A system chaing said to exibit backward chaining also called consequent driven, top-down,goal-driven, if it tries to support agoal stae or hyupothesis by checking known facts in the context. IF the facts i the cintext do not support the hypothesis, then the precinditions that are needed for he hypothesis ar set up as subgoals. Essentially, the process can be viewed as a search in teh sate space going from teh goal state to the initial staste by the application o inverse operators and involves depthj firat search. he concept of backward chaining may be applied to the decomposition of the tasks in engineering design. If the ucrrent state of the context is not in the proper form, for th completion of a task, the task may be decomposed into subtasks. In this way the ovreall design task may be decomposed into several subtasks and backward chaianing may be applied to each subtask.

Backtracking : The problem reduction approach is applicable to problems that can be subdivided into a tree of fixed sub-problems. However, in a number of practical problems, it may not be possible to decompose problems into a fixed set of problems. A number of alternative approaches may exist. In backtracking, the problem solver backs up to other nodes, at the same level of starting node, if no solution is found along the current path. Backtracking is inbuilt in AI language Prolog.

Means-ends analysis : In means-ends analysis, the difference between the current state and the goal state is determined

THE SUITABILITY OF PROLOG

The expert system is expected to contain knowledge, inference mechanism and context based decisions are to be taken, avoiding unnessasary search this mechanism cannot be implemented by the conventional languages like BASIC, FORTRAN, PASCAL. In the conventional languages the CPU works its way through a sequence of operations in a predefined way. It can make choices which allow it to follow different paths through the program but only those which the programmer has forseen. This fits well for a numerical problem which are easy to program and can be implemented manually by hand but highly unsuitable to implement and work with knowledge, which is represented by rules.

In conventional languages used for engineering and Scientific work the flow of control is predefined by the programmer. So the programme will be executed in the same sequence of steps. To implement more logical problems using conventional languages calls for a complicated IF THEN ELSE rules and loops, moreover the program is not extendable, additions to the programme is not easy to implement.

THE DISADVANTAGES OF THE CONVENTIONAL LANGUAGES

1) The program is executed in a predefined pattern defined by the author.
2) Addition to the program is not easy to implement.
3) Implementation of complex logic calls for complicated loops and IF THEN ELSE rules.

4) The state of variables change with time, so a statement means different depending when it is called.

5) Verifications of the program,to ensure the desired route is taken and the result is achieved is very difficult.

6) Intutive judgement cannot be implemented using conventional languages like BASIC, FORTRAN, PASCAL.

Civil engineering design problems characterised by their inherent imprecision, the paucity and incompleteness of data, and heavy reliance on expert views. Since experience and intutive judgement form and important role in the design process.

A language which can implement heuristic knowledge is needed to implement designing of civil engineering problems. Artificial intelligence languages like LISP, PROLOG are the promising languages to implement knowledge and symbolic computations.

SUITABILITY OF PROLOG TO BUILD EXPERT SYSTEMS

The PROLOG provides a uniform data structure called TERM, out of which all data as well as Prolog programs are constructed. A PROLOG program consist of a set of clauses where each clause is either a fact about the given information or a rule about how the solution may relate or be infered from a given facts. Thus Prolog can be viewed as first step towards the ultimate goal of "Programming in logic".

Prolog can be viewed as a discriptive language as well as a perspective one. The prolog approach is rather to to prescribe the sequence of steps taken by the computer to solve the problem. When a computer is programmed in Prolog, the actual way the computer carries out the computations is specified partly by logical semantics of prolog, partly what new facts prolog can infer from the given one and only partly be explicit controlling from information supplied by the programmer.

Imprecision underlines many of the fundamental principles of the subject. Hence our factor of safety, which are quite large because we do not know accurately the magnitude of some of the qaunatities we have to use.

Expert systems, particularly those writtren in Prolog, may allow us to incorporate imprecision into our design process directly, rather than indirectly. It is not suggested that all design could benefit from this approach, but in certain structural applications where an understanding of the probabilities of failure is required, it would certainly be of benefit.

Imprecision can be incorporated directly into our reasoning in the following way. Instead of the precise statement

P is true if Q is true and R is true

Prolog allows us to write an imprecise version of the same rule as

P is true with support x if
Q is true with suport y and
R is true with support z and
Y combined with z gives x

The last clause, which has been deliberately left vague, is used to define some relationship between the support quantities x, y, and z. It may be a clause which is specific to the problem in hand, but more likely, it will be a clause written in terms of a standard theory, such as probability or fuzzy logic. Expert systems have already been written in this way, for example the MYCIN project looking at medical diagnosis , and much work is both underway and remains to be done before practical applications can be made in our fields. Not the least of these is the problem of obtaining the basic data for the values of support in the first place.

PROLOG is a logic programming language. It has a sound mathematical basis, the first order predicate calculus. Wide range of world facts can be represented by first order predicate calculus, which can be implemented on a computer. The language predicate calculus consists of a number of components such as predicate symbols, variable symbols, function symbols and constant symbols. These features alone give program an edge over the other declarative languages such as PRUF and FRIL. Another extremely valid reason for promoting for prolog for expert systems is the expected revolutionary changes to come due to Japanese Fifth generation project (FGCS). The fifth generation languages will be capable of exploring the highly parallel computation architectures to come. This architectures support many central processing units and each CPU simultaneously executes the subdivided problem, which are independent, not sharing the same data or the input is ready from other CPU for it by the time starts execution.

Prolog can efficiently support the use of parallel processing.

The inference engine in Prolog is essentially a theorem prover, which tries to prove the goal by proving each of the subgoals starting from the leftmost subgoal in a depth first manner. Hence, the depth first strategy is built into the control mechanism. However, other problem solving strategies can be easily programmed. If any of the subgoals are not satisfied for a particular binding of variables, then the system backtracks and the program continues with a new set of variable bindings. These variable bindings are available in the database; the variable bindings can also be provided through some user defined functions.

TOOL/LANGUAGE	DEVELOPER	REPRESENTATION SCHEME	IMPLEMENTATION LANGUAGE
OPS5	CMU	Rules	LISP/BLISS
EMYCIN	STANFORD	Rules	INTERLISP
HEARSEAY-III	USC-ISI	Rules	INTERLISP
EXPERT	RUTGRS	Rules	FORTRAN
ROSIE	RAND CORP.	Rules	INTERLISP
KS1ØØ	TECKNOWLWDGE	Rules	INTERLISP
AGE	STANFORD	Rules	INTERLISP
KAS	SRI Int.	Rules	
KMS	MARYLAND	Rules/Frames	LISP
KEE	Intelligentics	Rules/Frames	INTERLISP
R11	STANFORD	Frames	INTERLISP
PSR1	CMU	Frames	INTERLISP
LOOPS	XEROX-PARC	Rules&Frames	INTERLISP
C PROLOG	Edinburgh	Logic	C

ADVANTAGES OF USING PROLOG

1) Prolog is very powerful in manupulating expressions.
2) It has an inbuilt depth first inference engine.

3) Prolog programmes are capable of modelling to some extent the human cognitive ability.

4) Prolog is very dynamic language it can be easily extended.

5) Prolog is easyly readable.

6) Prolog programes are very easy to debug using the built in "Trace" predicate.

7) Recurssion using lists, Prolog can simulate all the functions available in the numerical computation language.

8) Prolog programming can be data directed style, this feature makes prolog dynamic and easy to change.

9) Fuzzy logic and imprecision data can be used in Prolog,

Develoment of ES requires flexibility and case of modification. The procedural interpretation of Prolog makes the Prolog program entirely modular. Consider the following set of knowledge-basic rules.

R : if B and C are true. then A is true.

R : if D and E are true. then B is true.

The equivalent Prolog clauses :

C : A if B and C

C : B if D and E

Note that the Prolog clauses are in fact almost identical to the natural language rules themselves except that the conclusion precedes the conditions. The two rules exist as independent Prolog clauses. Each clause can be mdified or used as an independent procedure. The independent status of clauses makes the Prolog program entirely modular. Modularity of programs coupled with the interpretive implementation of most Prolog systems makes them easy to modify.

ES programs must be able to explain their line of reasoning and Prolog fulfils this requirement by providing an inbuilt inference system a Prolog shell. ESs are develoed incrementally through modification and refinement cycles. This increases the possibility of inconsistencies creeping into the data and knowledge-base. Being a purely logic-based system. Prolog has a much better chance of being able to detect these inconsistencies. Consider the following set of Prolog clauses:

```
C1: A  if  not  B
C2: B  if  C
C3: A  if  C
C4: C
```

C4 states that C is true from C2 and C3, it therefore follows that A and B are true. However, C2 states that A is true only if B is not true. Thus the set of clauses stated are inconsistent, which Prolog will immediately spot. Prolog does not make any distinction between data and knowledge, but providies a uniform formalism for the representation of both the data and knowledge-bases.

RULES IN THE KNOWLEDGE BASE

The Knowledge base of the expert system contains the knowledge in the forms of rules or in the form of data. This expert system is furnished with the following rules which are of the form of rules and data. The design of the beams strictly follows IS-456. the rules which are relevent and incorporated in the program are listed below.

4.2 Effective span

Simply suported Beam :The effective span of a member that is not build integraly with its support shall be taken as clear span plus the effective depth of slab or centre to centre of suppports, whichever is less.

Continuous Beam : In the case of continuous beam, if the width of the support is less than 1/12 of the clear span, shall be as above. If the supports are wider than 1/12 the clear span or 600 mm whichever is less, the effective span shall be taken as under.

1. For the span with one end fixed and the other continuous or for intermediate spans, the effective span shall be the clear span between supports and

2. For end span with one end free and the other continuous, the effective span shall be equal to the clear span plus half the effective depth of the beam or slab or the clear span plus half the width of the discontinuous support, whichever is less.

4.4.1 Arrangement of live loads:

a)Consideration may be limited to combinations of:

1.Design dead loads on all spans with full design live loads on two adjacent spans, and

2. Design dead loads on all spans with full design live loads on alternate spans.

b) When design live load does not exceed three-fourth of the design dead load the load arrangemets may be design dead load and design live load on all the spans.

22.2 Control of deflection:

The deflection of a structre or part there off shall not adversly affect the appearance or efficiency of the structure or finishes or partitions.

The deflections shall genarally be limited to the following:

a) The final deflection due to all loads including the effects of temperature, creep and shrinkage and measured from the as-cast level of the supports of floors, roofs and all other horizontal members, should not exceed span/250.

b) The deflection including the effects of temperature, creep and shrinkage occurring after erection of partitions and the application of finishes should not normally exceed span/350 or 20 mm whichever is less.

4.2.1 For beams and slabs, the vertical deflection limits may generally be assumed to be satisfied provided that the span to depth ratios are not greater than the values obtained as below:

a) Basic values of span to effective depth ratios for spans up to 10 meters:

cantilever	7
Simply supported	20
Continuous	26

b) For spans above 10 m, the values in (a) may be multiplied by 10/span in metres, except for cantilever in which case deflection calculations should be made.

c) Depending on the area and the type of steel for tension reinforcement, the values in (a) or (b) shall be modified as per Fig. 4.1.

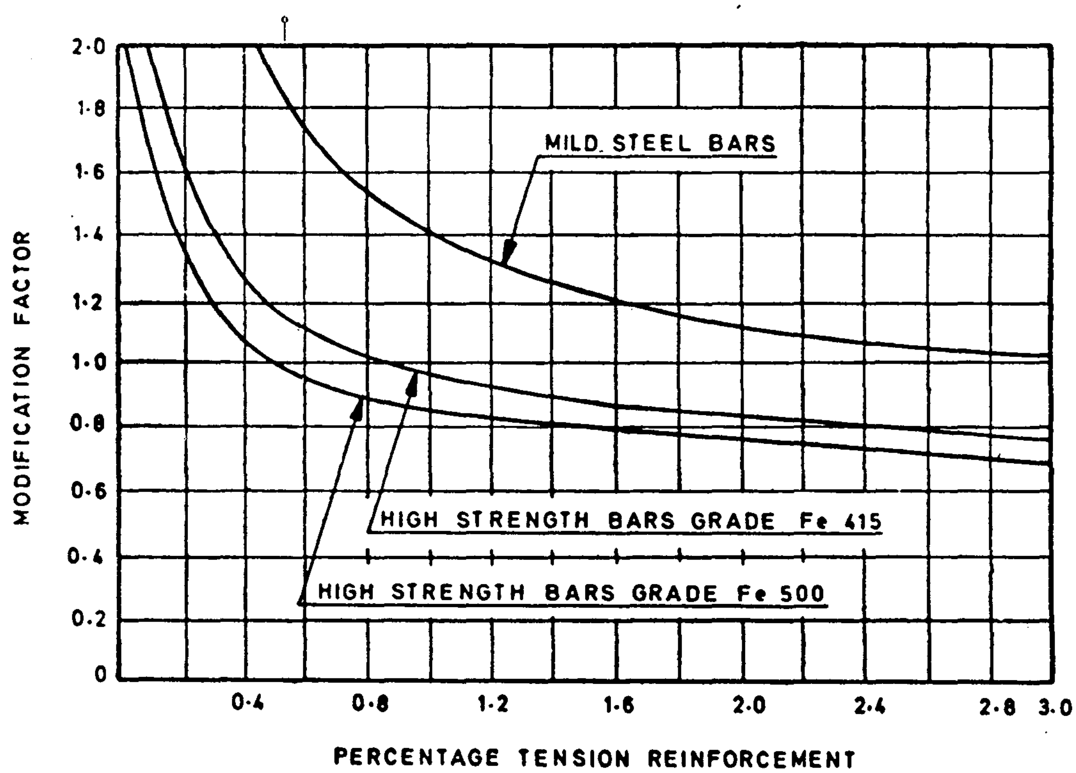


FIG.4.1

d) Depending on the area of compression reinforcement, the value of span to depth ratio be further modified as per Fig. 4.2.

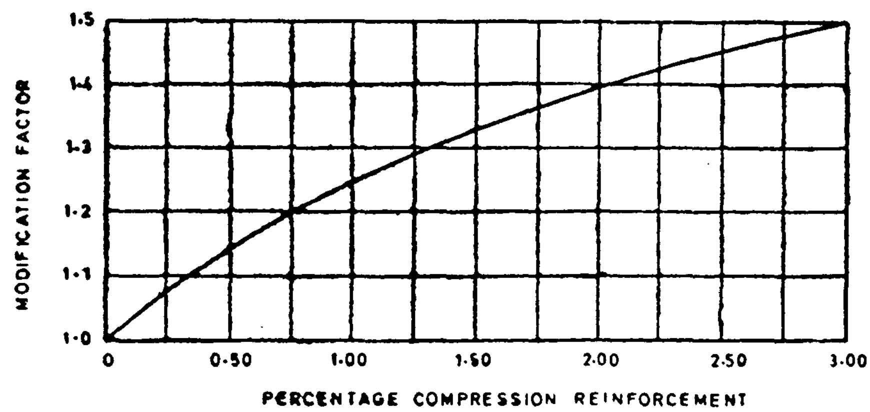


FIG.4.2.

4.3 Slenderness Limits for Beams to Ensure Lateral Stability

A simply supported or continuous beam shall be proportioned that the clear distance between the lateral restraints does not exceed 60 b or 250 b**2/d whichever is less, where 'd' is the effective depth of the beam and b the breadth of the compression face midway between the lateral restraints.

For a cantiliever, the clear distance from the free end of the cantilever to the lateral restraint shall not exceed 25 b or 100*b**2/d whichever is less.

REQUIREMENTS GOVERNING REINFORCEMENT.

General: Reinforcing steel of same type and grade shall be used as main reinforcement in a structural member. However simultaneius use of two different types of grades of steel for main and secondary reinforcement respectively is permissible.

Bars may be arranged singly or in pairs in contact, or in groups of three or four bars bundled in contact. Bundles shall not be used in a member without stirrups. Bundled bars shall be tied together to ensure the bars remaining together. Bars larger than 36mm diameter shall not be bundled, except in columns.

Development of stress in reinforcement :

The calculated tension or compression in any bar at any section shall be developed on each side of the section by an appropiate development length or anchorage or by a combination thereof.

Development length of bars: The development length Ld is given by Ld = Phi*S/(4*T')

where Phi = The nominal diameter of the bar.

S = the stress in the bar at the section considered at the design load, and

T' = design bond stress given below.

Design bond stress in limit state method for plain bars in tension shall be as below.

Grade of concrete	M15	M20	M25	M30	M35	M40
Design bond stress	1.0	1.2	1.4	1.5	1.7	1.9

For deformed bars confirming to IS:1786 -1979 or IS:1139

1966.

These values shall be increased by 60% for bars in compresssion the values of bond stress for bars in tension shall be increased by 25% .

Bars bundled in contact: The development length of each bar of bundled bars shall be that for the individual bars, increased by 10% for two bars in contact, 20% for three bars in contact , and 33% for four bars in contact.

Anchoring reinforcing bars in tesion :

a) Deformed bars may be used without end anchorages provided the development length requirement is satisfied. Hooks should normally be provided for plain bars in tesion .

b) Bends and hooks shall confirm to IS-2502-1963

1) Bends : The anchorage value of bend shall be taken and four times the diameter of the bar for each 45 degrees of bend subject to a maximum of sixteen times the diameter of the bar.

2) Hooks: The anchorage value of a standard U type hook shall be equal to 16 times the diameter of the bar.

Anchoring bars in compression : The anchorage length of straight bars in compression shall be equal to development length of bars in compression as specified above. The projected length of hooks, bends and straight length beyond bends if provided for a bar in compression, shall be considered for development length.

Anchoring shear reinforcement :

a) Inclined bars: The development shall be as for bars in tension this length shall be measured a under:

1) In tension zone, from the end of the sloping or inclined portion of the bar and

2) In the compression zone in the mid depth of the beam.

b) Stirrups : Not withstanding any of the provision of this standard in case of secondary reinforcemnt such as stirrups and transverse ties complete development length and anchorage shall be deemed to be priovided when the bar is bent to an angle of at least 90 degrees round a bar of atleast of its own diameter and is continued beyond the end of the curve for a length of atleast 8 diameters or the bar is bent through an angle of 135 degrees and is continued beyond the end of curve for atleast six bar diameters or the bar is bent through an angle of 180 degrees and is continued beyond the end of curve for a length of atlest four bar diameters .

Bearing stresses at bends : The bearing stress in concrete for bends and hooks described in IS-2502-1963 need not be checked. The beaing stress inside a bend on any other bend shall be calculated as given below.

Bearing stress = Fbt/(R*Phi)

where Fbt = Tensile force due to design loads in bar or group of bars,

R = Internal radius of the bend and

Phi = size of the bar or, in bundle, the size of bar of equivalent area.

For limit state method of design this stress shall not exceed 1.5*Fck/(1+2*Phi/a) where Fck is the characteristic strength of concrete and 'a', for a particular bar or a group of bars shall be taken as the center to centre distance between bar of the group of bars perpendicular to the plane of the bend; for a bar or a group of bars adjecent to the face of the member 'a' shall be taken as the cover plus size of bar (Phi).

If a change in direction in tension or compression reinforcement induces a resultant force acting outward tending to split the concrete, such force should be taken up by additional links or stirrups. Best tension bar at a reentrant angle should be avoided.

Curtailment of tension reinforcement in flexural members:

For curtailment, reinforcement shall extend beyond the point at which it is no longer requried to resist the flexure for a distance equal to the effective depth of the member or twelve times the bar diameter, whichever is greater except at simple support or end of cantilever.

a) The shear at the cut-off point does not exceed two-thirds than permitted, including the shear strength of web reinforcement provided.

b) Stirrups area in excess of that required for shear and torsion is provided along each terminated bar over a distance from the cut-off point equal to three-fourths the effective depth of the member. The excess stirrup area shall not be less than Ø.4*b*S/f,where 'b' is the breadth of beam,'S' is the spacing and 'f' is the characteristic strength of reinforcement in N/mm. The resulting spacing shall not exceed d/8 where 'd'is the ratio of the area of the bars cut off to the total area of bars at the section, and d is the effective depth.

c) For 36mm and smaller bars, the continuing bars provide double the area required for flexible at the cut-off point and the shear does not exceed three-fourths that permitted.

Positive moment reinforcement

a) At the one-third the positive moment reinforcement in simple members and the one-fourth the positive moment reinforcement in continuous members shall extend along the same face of the member into the support, to a length equal to:

b) When a flexural member is part of the primary lateral load resisting system, the positive reinforcement required to be extended into the support as described in (a) shall be anchored to develop its design stress in tension at the face of the support.

c) At simple supports and at points of inflection, positive moment tension reinforcement shall be limited to a diameter such that computed for by 4.2.1 does not exceed

where m1 = moment of resistance of the section assuming all reinforcement at the section to be stressed to:

f = 0.87 in ther case of limit state design and the permissible stress in the case of working stress design;

V = shear force at the section due to design loads;

l = sum of the anchorage beyond the centrer of the support and the equivalent at simple support, and at a point of inflection, is limited to the effective depth of the members or 12, whichever is greater.

Negative moment reinforcemnet: At least one-third of the total reinforcement provide for negative moment at the support shall extend beyond the point of inflection for a distance not less than the effective depth of the member or '12' or one-sixteenth of the clear span whichever is greater.

Curtailment of bundled bars-Bars in a bundle shall terminate at different points apart by not less than 4Ø times the bar diameter except for bundles stopping at a support.

Special members-Adequate end anchorage shall be provided for tension reinforcement in flexural members where reinforcement stress is not directly proportional to moment, such as sloped,stressed or tapered footings, brackets, deep beams, and members in which the tension reinforcement is not parallel to the compression face.

Reinforcemnet splicing- Where splices are provided in the reinforcement bars, they shall as far as possible be away from the sections of maximum stress and be staggered. It is recommended that splices in structural members should not be at sections where the bending moment is more than 5Ø% of the moment of the resistance, and not more than half the bars shall be spliced at a section.

Where more than one half of the bars are spliced at a section are where splices are made at points of maximum stress, special precautions shall be taken, such as increasing the length of lap and/or using spirals or closely spaced stirrups around the length of the splice.

4.3 Spacing of reinforcement :

4.3.1 Minimum distance between individual bars :

a) The horizontal distance between two parallel main reinforcement bars shall usually not less than the greatest of the following -

1. The diameter of the bar, if the diameters are equal.
2. The diameter of the larger bar if the diameters are unequal and
3. 5mm more than the nominal maximum size of coarse aggregate.

b) Greater horizontal distance than the maximum specified in (a) should be provided wherever possible. However when needle vibrators are used, the horizontal distance between the bars of a groove may be reduced to two thirds the nominal maximum size of the force aggregate, provided that sufficient space is left between groves of bars to enable the vibrator to be immersed.

c) Where there are two or more rows of bars, the bars shall be vertically in line and the minimum vertical distance between the bars shall be 15mm, two thirds the nominal maximum size of aggregate of the maximum size of bar, which ever is geratest.

4.3.2 Minimum distance between bars in tension - Unless the calculation of crack widths shows that a greater space is acceptable, the following rule shall be applied to flexural members in normal internal and exrenal conditions of expose.

a) Beams :The horizontal distance between parallel reinforcement bars or groves, near the tension face of the beam shall not be greater the value given in the table below depending on the amount of redistribution carried out in analysis and the characteristic strength of the reinforcement.

Fy	Percentage redistribution				
	-30	-15	0	15	30
	Clear distance between bars				
N/mm^2	mm	mm	mm	mm	mm
250	215	260	300	300	300
415	125	155	180	210	235
500	105	130	150	175	195

Table 4.4

Cover to reinforcement

4.4.1 Reinforcement shall have concrete cover and thickness of such cover excluding the plaster and other decorative finish shall be as follows

a) At each end of the reinforcing bar not less than 25mm, nor less than twice the diameter of such bar.

b) For longitudinal reinforcing in a beam not less than 25mm nor less than the diameter of such bar.

c) For any other reinforcement not less than 15mm, nor less than the diameter of such bar

4.4.2 Increase cover thickness may be provided when surface of concrete members are exposed to the action of harmful chemicals, in such increase of cover may be between 15mm and 50mm beyond the figures given in 4.4.1 . 4.4.2.1 For reinforcement concrete members, periodically immerced in s water are subject to sea spray, space the cover of concrete shall be 5Ømm more than that specified in 4.4.1

4.4.3. For concrete of the grade M25 or above the additional thickness of cover specified in 4.4.2 to 4.4.2.1 may be reduced to half. In all such cases the cover should not exceed 75mm.

4.5 Requirements of reinforcement for structural members

4.5.1 Beams

Minimum tension reinforcement : The minimum area of tension reinforcement shall not be less than that given by the following

$$As/bd = 0.85/Fy$$

where As = minimum area of tension reinforcement

b = breadth of the beam

d = effective depth

Fy = characteristic strength of reinforcement

b) Maximum tension reinforcement : The maximum area of tension reinforcement shall not exceed Ø.Ø4 * b * D

Compression reinforcement : The maximum area of compression reinforcement shall not exceed Ø.Ø4 * b * D. Compression reinforcement in beams shall be enclosed by stirrups for effective lateral restraint.

Side face reinforcements : Where the depth of the web exceeds 75Ømm in a beam, side face reinforcement shall be provided along the two faces. The total area of such reinforcement shall be not less than Ø.1% of the web area and shall be distributed equally on two faces at a spacing not exceeding 300mm or web thickness whichever is less.

Transvers reinforcement in beams for shear and torsion : The transfer reinforcement in beams shall be taken around the outer most tension and compression bars.

Maximum spacing of shear reinforcement : The maximum spacing of shear reinforcement measured along the axis of the member shall not exceed 0.75*d for vertical stirrups and 'd' for inclined stirrups at 45 degrees, where 'd' is the effective depth of the section under consideration. In no case shall the spacing exceed 450mm.

Minimum shear reinforcement: The minimum shear reinforcement in the form of stirrups shall be provided such that

$$Asv/(b*Sv) >= 0.4/Fy$$

where Asv = total cross-sectional area of stirrups legs effective in shear.

sv = stirrup spacing along the length of the member,

b = breadth of the beam or breadth of the web of flanged beam, and

Fy = characteristic strength of the stirrup reinforcement in N/mm which sahll not be taken greater than 415 N/mm*mm.

However, in members of minor structural importance such as lintels or where the maximum shear stress calculated is less than of the permissible value.

Distribution of torsion reinforcement: When a member is designed for the torsion reinforcement shall be provided as below.

a) The transverse reinforcement for torsion shall be rectangular closed stirrups placed parpendicular to the axis of the member. The spacing of the stirrups shall not exceed the least of the X1, (X1+Y1)/4 and 3ØØ mm, where X1 and Y1 are respectively the short and long dimensions of the stirrups.

THEORY OF DESIGN

The expert system implements the design by 'Ultimate flexural strength' design theory,also called as 'Limit state' of design.The following asssumptions were made in the ultimate flexural strength design. 1. Plane sections normal to the plane of bending remain plane after bending.

2. The strain in concrete at the outermost compression fibre reaches a specified value only at failure.

3. The distribution of compressive stress in concrete at failure is defined by an idealised stress strain curve.

4. The tensile stress in concrete is totally ignored.

5. The stress in the reinforcement is derived from representative stress strain diagram of the steel used.

The first assumption stipulates linear strain profile across the depth, that is the strains in concrete and reinforcement are directly proportional to the distance from neutral axis at which the strain is zero.The ultimate strain in concrete varies between wide limits. The ultimate flexural strength is not appreciably influenced by it. A value of the order of 0.003-0.0035 is generally adopted for the purposes of design.

The stress-strain relation recommended in the Code is shown in Fig.5.1 with the maximum compressive stress at 0.67 Fck (Fck=Fcu) .The compressive stress distribution in the beam is defined as the stress block and can be readily traced as shown in Fig.5.2 in which X=Ecu*D/(Zcu+E8) from similar triangles and O in Fig.5.1 corresponds to the neutral axis.

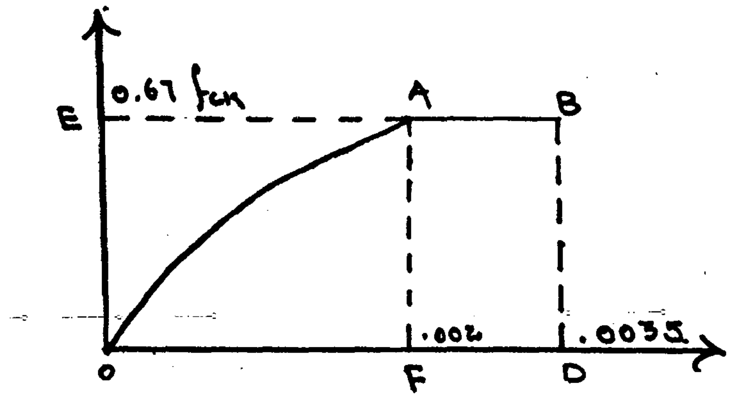


Fig 5.1

The total compression C = B * Area OABD

and the moment of resistance Mu= C(d-K2*X)

where K2*X is the distance of the centroid of the stress block from the outermost compression fibre.

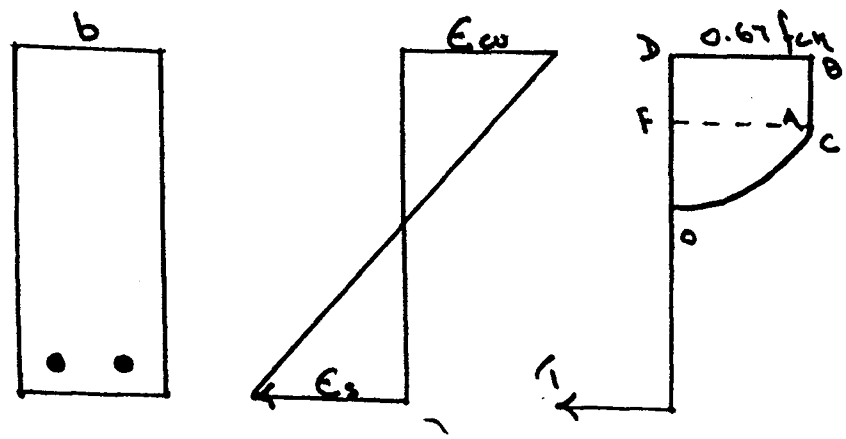


FIG.5.2

The actual shape of the stress block is not important and in fact the Code permits rectangular, parabolic or other shapes to be used which provides reasonable agreement with test results. The equivalence can be easily established. We can deduce the equivalent rectangular stress block for the parabolic shape in Fig.5.3b.

With d-Ø.375*X = d-Ø.5*a,we get the hypothetical depth

$a = 0.75*X$ in Fig.5.3c. Again letting Cp = Cr, we get

$$2*A'*Fck*b*X/3 = B'*Fck*b*a$$
$$B' = 2*X/(3*a) = 8*X/9.$$

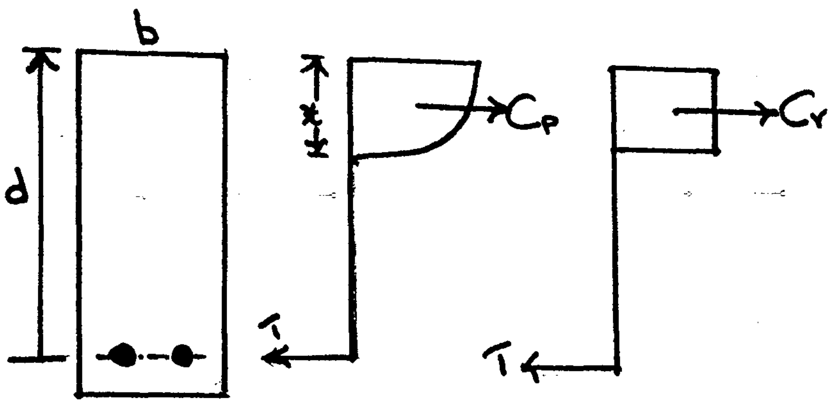


FIG .5.3

The depth 'a' is not the depth of the neutral axis. The stress block is very conveniently expressed with the aid of the following three parameters.

K1 = Area OABD/Area OEBD in Fig.5.1, and

k2 = Area OABD/Area(x.DB) in Fig.5.2,

The values of K1 and K2 for Fig.5.1 are derived in the following:

$$AreaOABD = AreaOAF + AreaABDF = 2*4*OD*BD/(3*7)+3*OD*BD/7$$
$$= 17*Area\ SOEBD/21,$$
$$K1 = 17/21 = 0.8095 = 0.81.$$
$$K2 = 0.416=0.42.$$

The compressive force in Fig.5.2

$$C=K1*K3*Fck*b*X \qquad 2.1$$

The moment of resistance

$$Mu = C(d - k2*X) \qquad 2.2$$

$$= K1*k3*Fck*b*d*d(x/d)(1-0.42(x/d))$$

The choice of k3 =0.67 in the Code provides a factor of safety of 3.36 against failure in concrete as illustrated in Section 11.2.2.2.

Recommendations of IS-456

The general expressions, 5.1 and 5.2 are readily modified for design by incorporting the partial safety factors Gms and Gmc. The derivations are summarised with reference to Figs 5.4 and 5.5 using rms=1.15, rmc=1.

k1=17/21, k2=Ø.42 and k3 =Ø.67.

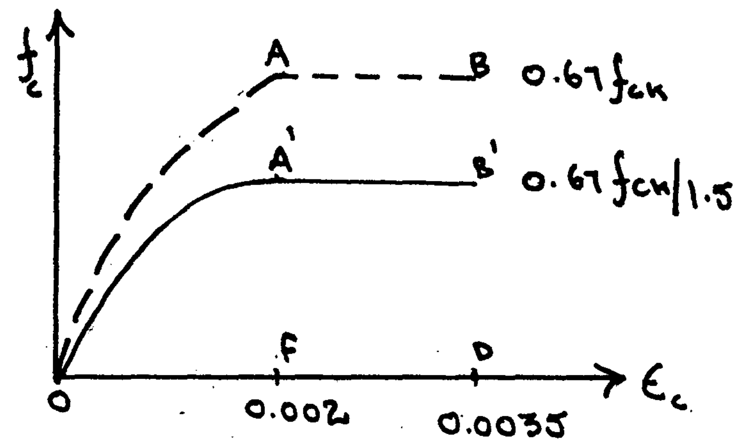


FIG.5.4.

The Code stipulates that at failure the strain in reinforcement should not be less than Es=Ø.ØØ2+(Ø.87Fy/Es) thereby ensuring a stress of Ø.87*Fy in Fig. 5.7

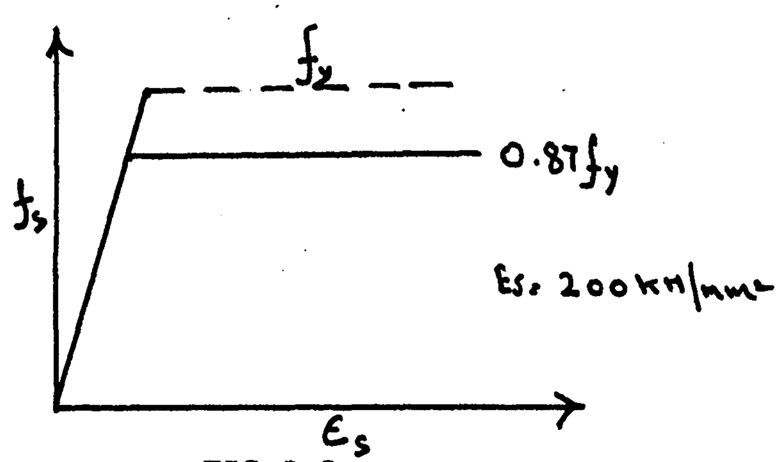


FIG 5.5

Referring to Fig.5.6, the maximum value

X1 um = (Xu/d)max = Ø.ØØ35*Es/(Ø.ØØ55*Es + Ø.87*Fy).

$$= 805/(1265+Fy) \qquad --- \qquad 5.3$$

The compressive force in concrete

$$Cc = 17*0.67*Fck*b*Xu/(21*1.50)$$
$$= 0.36*Fck*b*Xu$$

The derivations for the moment of resistance of rectangular section is given below.

RECTANGULAR BEAM WITH TENSION REINFORCEMENT

Assigning Asc = 0, equilibrium of forces in Fig.5.6c

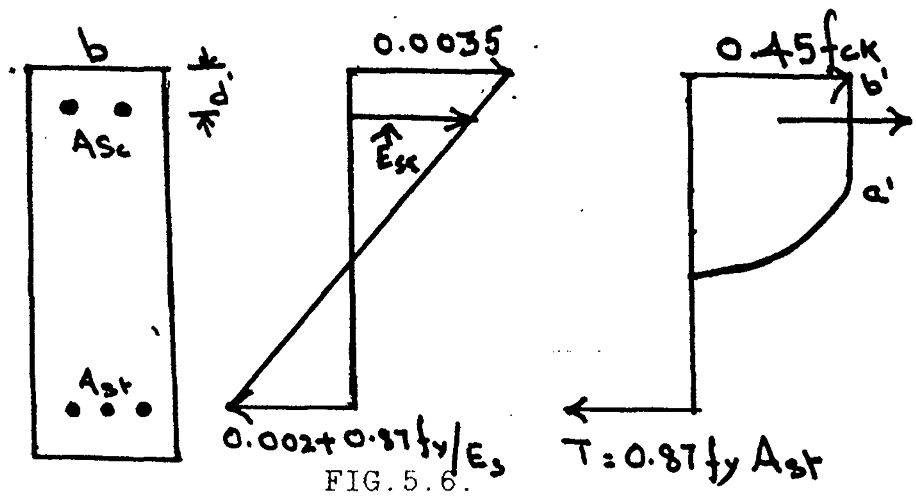


FIG.5.6.

$$Cc = T$$
$$X' = Xu/d = 0.87*Fy*Ast/(0.36*Fck*B*D)$$
$$= 2.417*P*Fy/Fck$$

where p = Ast/bd and the limiting value of X', X'lim is given by eq.5.3. Three possible cases are examined.

Case 1: X' < X'lim

For this case the moment of resistance is conveniently expressed as

$$Mu = Td(1-0.42*X')$$
$$= 0.87*Fy*p*b*d*d(1-(0.42*2.417*p*fy/Fck))$$
$$= 0.87*Fy*p*b*d*d(1-(1.015\ p*fy/Fck))$$
$$= 0.87*Fy*p*b*d*d(1-pfy/Fck)$$
$$= 0.87*Fy*Ast(d-(Fy*Ast/Fck*b))$$

Case 2 : $X1 = X1 \lim$

Compression $Cc = 0.36*Fck*b*d*X'\lim$ and

Tension $T = 0.87*Fy*b*d*plim = Cc$

The moment of resistance

$$Mulim = 0.36*Fck*b*d*d(1-(0.42*X'\lim)X'\lim \qquad -- 5.4$$

is the limiting value and corresponds to the balanced state in Working Stress Method

Case 3 : $x1 > x1 \lim$

For this case $P > Plim$ and the section should be redesigned for economy implying thereby that increase over Mulim is not permitted for $P > Plim$.

RECTANGULAR BEAM WITH COMPRESSION REINFORCEMENT

Such reinforcement would be required when the applied moment 'Mua' on a section is larger than Mulim given by eq.5.4.

The difference is carried by additional tensile reinforcement Ast, compression reinforcement Asc.

Referring to Fig.5.6b, the strain in compression reinforcement $Esc = 0.0035(Xu\ max - d')/Xumax$ for which let the corresponding stress, read off from the stress-strain curve, be denoted by Fsc.

$$Asc = Mua - Mu\ lim /Fsc\ (d-d1)$$

$$Ass = Mua/0.87*Fy - Mulim /(d-d1) = Asc\ Fsc/0.87*Fy$$

The total tensile reinforcement is given by

$$Ast = Plim*b*d + AS2$$

The presence of Asc is neglected in the computation of Mulim as the modification is not appreciable.

IMPLEMENTATION OF ANALYSIS IN PASCAL

The implementation of the expert system to design beams and structures is two fold, one is the analysis and the other is the design. First the beam has to be analyzed and the moments produced by the loads are computed which is called the analysis. In the design, the section which can safely handle the loads in the worst circumstances the structure is most likely to be submitted.

All the analysis of the beam is implemented in the programming language Pascal and the designing, which is more logical and complicated in which the need of the heuristics arise is implemented in Prolog. The designer must opt for only one section, where there can be infinite sections which can satisfy the moments induced. Here the experience and efficiency is demanded to opt for the best of all satisfing the constraints specified.

I confined myself to the problem of analyzing the beams of four types:

1. Cantilever beam
2. Simply supported beam
3. Over hanging beam and
4. Fixed beam

Two types of loadings 1.Point load and 2.Uniformly distributed load were considered. In the analysis of the beams, there can be any number of loads either point load or UDL, and the moments induced by them can be precisely computed. The section of the beam which can safely handle the loads following IS-456 specifications is decided in the designing part of the progaram which is implemented in Prolog.

ANALYSIS

In the analysis of the beam, it is necessery to compute the moments induced by the loads at every point of length of the beam. So the beam is divided into 100 equal parts which can be represented by an array of dimension 100 and the corresponding moments and forces are stored in the array. The advantage of using the array is that, the moment at any point on the beam can be accessed dynamically by accessing the corresponding array position at any stage of execution. Instead of using the arrays, packed arrays are used to save the memory space of the computer.

All the computations and variable values are stored and retrieved by using the array C of dimension 10x100. This array C[i,1] where i varies from 0 to 100, contains values representing the whole length of the beam. Each segment contains the value of 1/100 of length, i.e. if the length of the beam is 12 metres, then C[0,1] is assigned zero, C[1,1] contains 12 and C[2,1] contains 24 and so on. So the array C[i,1] contains the length of the beam in centimeters. Proper care has been taken in the computations to change this units to meters. C[i,2],C[i,3],C[i,4] contain the intermediate values and C[i,5] contains the final bending moments induced by a single load. Many variables, mostly real, strings which read the answers for the queries and flags which are boolean and text files to write the output were used in the program. All the variables are assigned to zero.

The flag OVER which initiates the major while loop in the program is set to true.This flag is false if the user responds N to the question "Any other loads[y/n] ".The loads are analized one by one and finally added to get the final moments induced. When this ANALYSIS.PAS is called by the DESIGN.PRO ,the query "Which type of beam you like to design" appears on the screen. The user is required to enter "cl", "ss", "oh", "fx" to design a cantilever beam, simply supported, over hanging and fixed beams respectively.The responsed of the user is stored in BMTYPE. If the user by mistake opts for none of these, he will be prompted again.It is assumed that the beam is horizontal and all the loads are acting in the downward direction and the supports in the upward direction. This message is displayed on the screen and the user is requested to enter the loads on the beam one after the other. There are two options available point load and UDL. The user is requested to enter "1" for point load and "2" for the UDL.The first option sets the PTFLAG to TRUE which will later initialize the routines which analyze the point load and the second option sets the UDL flag to true. Now the program splits into two parts, one for calculating the moments and the other for calculating the shear forces for all the four types of beams.

The grade of the steel the user is going to use for the reinforcement has to be specified by the user. If not specified appropriate values are assumed as per IS-456. Three grades of steel FE-500, FE-415 and FE-250 which are HYSDsteel, ribbed tar steel and mild steel respectively. The user is requested to enter 1,2 or 3 to opt for the steel specified as above. If the grade of steel selected is found unsatisfactory later in the design, the user is requested to change the option simultaneously advising the best suited. If the user still insists on using the same grade of steel, other possible changes satisfying the constraints are deemed. Next the user is requested to enter the grade of concrete he prefers ranging from M10 to M40. The user can not opt other than these. Next the user is requested to enter the length of the beam in meters, which is assigned to the variable LENGTH. Depending on the option for the beam type, four separate routines were provided to fix the end conditions. Except in the case of over hanging beam, for all the other three, the support conditions are x=0 and y=length where x is the left support and y is the right support measured from the left. x should never be less than zero and y not greater than the length. A small routine is provided to check these conditions.

If FXflag is true, this flag is assigned true if the user designs a fixed beam, x is assigned zero and y length. It is necessery in case of fixed beam to check the sinking of the supports, which may induce large amount of moments. If any of the supports sink due to construction of foundation inaccuracies, they induce extra moments other than the loads. The user is questioned whether there is any sinking of supports. If the response is 'N' , the analysis of loads starts. If not,a routine which computes the moments induced due to sinking is run.

The user is requested to enter the values of 'E' and 'I' and to specify left or right support which has sunk and also the sinking in centimeters which is assigned to SNK. The moment induced

S1 = 6 * E * I * SNK / (LENGTH * LENGTH)

At both the ends equal magintude of moment is induced, but in opposite direction. The moment at the support which sinked down is assumed as positive and the other as negative. Since the moment changes linearly from positive to negative, it is easy to implement. The first array segment corresponding to the one end of the beam is assigned the moment computed before i.e S1. The last segment also assigned S1 with the opposite sign. All the intermediate values are computed by linear interpolation and saved in the array H[i].

```
S1 =  6 * E * I * SNK / (LENGTH * LENGTH)
S2 = S1 / 50 ; z = S1
FOR i = 1 to 100 DO
   BEGIN
        H[I] := Z;
        Z := Z - S2;
   END
```

The sinking moment is calculated and stored in the intermediate array H[i]. Later it is added to the final moment induced by the loads.

The flag ENTRY is set to true which initiates the major loop. All the intermediate values ,arrays are assigned zero. The array contains the values calculated for a single load and to arrive at the final moments all of them are added. Immediately after entering the major while entry loop, assuming at least one load on the beam, the entry falg is set to false. This falg will be assigned true only if tyhe user responds 'Y' to the question 'any other loads'.Now the user is requested to enter 1 or 2 to opt for point load or UDL. The user is requested to enter the location of the point load which is assigned to 'distance' and the magnitude od the load assigned to ' intensity'.

CANTILEVER BEAM

A message is printed that the cantilever is assumed to be fixed at the left end. We need such minor restrictions, Otherwise the user has to enter more data. The shear force induced by the load

Sh = Intensity

moment m1 = distance * intensity

The moment is zero from the right end upto the location of the point load and linearly increases to m1 upto support. Since the variation is linear it is easy to implement using arrays and interpolation.

C[i,5] contains the bending moment values and C[i,6] contains the shear force values.

OVERHANGING BEAM

Simply supported beam is assumed as the over hanging beam with no over hanging spans and a fixed beam is assumed as as simply supported beam with fixed ends, the fixed end moments are separately computed and later super imposed. A single routine developed to analize over hanging beam can be used for simply supported beam as well as fixed beam. The location of the point load can be

1. In between the supports and
2. On the over hanging span.

In case 1, no bending moment is induced beyond the supports. The shear force induced in the supports is the magnitude of the force itself but in opposite signs at the two supports. Implementation is quite simple. The value of intensity is stored beginning from the location where the load is located to the support. The same is repeated on the other side with opposite sign.

If a = distance between the load and the support

l = distance between the supports

b = l-a

i = magnitude of the load

then the momebt induced just beneath the load

$$m = i * a * b / l$$

and the moment at both the supports is zero. The implementation is done by linear interpolation.

UDL

If the user responds with '2', opting for the analysis of the UDL, the user is requested to enter the beginning of the UDL from left and this is assigned to 'distance' and the point wherte the UDL ends and this is assigned to 'span' and the intensity of UDL to 'intensity'.

Cantilever with UDL

To analize the cantilever loaded with UDL, let us consider the complicated case of the UDL spanning as shown in the figure above. The shear force is zero between 'b' and 'c' and parabolically increases between 'b' and 'a' and linearly increasesd between 'a' and support. Numerically

Shear force at b : Ø
Shear force at a : (a - b) * intensity
Shear force at support : (a - b) * intensity
The bending moment
between b and c : Ø
At any point between b and a :
intensity * (b - x) * (b - x) / 2
a and support :
i * (b - a) / 2 + ((a -x)+(b-a)/2)

where x is the point at which moment is computed,measured from the left.

The computation of the moment between 'a' and 'b' is facilitated by FOR loop accessig from 'a' to 'b'.The coresponding array location to 'a' is assigned

i * (S2-distance) * (S2-distance)/2 and

S2 is decremented by length/1ØØ each iteration.The value computed is assigned to the immediate next array location representing the moment at that point.

To computet the moment from support to a the same procedure of using a for loop is used. The location corresponding to the support is assigned

i * b - a *(distance between center of UDL and location where moment is computed). The x is decremented by length/1ØØ and this value thus obtained to the immediate next location and so on. To compute the shear force induced in the beam, one part computing between 'b' and 'a' and other 'a' and support. A for loop accessing from a to b is defined as

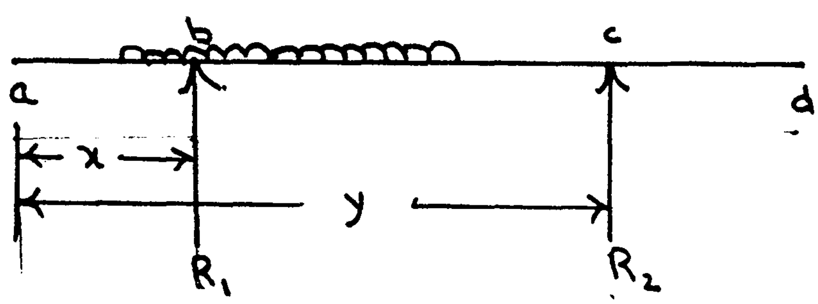
```
FOR i := trunc(round(distance*1ØØ/length))  to
         trun(round(span *1ØØ/length))      do
```

Array locations corresponding to distance to span one after the other. The computation is done in this loop and assigned the corresponding array location systematically.

Simply supported beam : This beam is analyzed as a overhanging beam with no over hanging spans. If the user opts for a simply supported beam, the same routine analyzing overhanging bean is called with the value of x and y assigned to Ø and distance.

Over hanging beam : The analysis of over hanging beam with UDL is the most complicated case of all the cases. To compute the shear force, the problem is subdivided into three parts for clarity.

case1 : Span <= y



FIG

The first step is to calculate the reactions of the supports 'b' and 'c'. In this case of 'span <= y' the moments about 'c' were computed. For example, considering the moments about 'c'

```
i*(c-a)*(c-a)/2+b*(c-b) = Ø
b = i * (c-a) * (c-a) / 2 / (c-b) = R1
R2 = (Intensity * ( C-A )) - R1
```

To compute the shear force over the nbeam 3 arrays C[I,6], C[i,7], C[i,8] were reserved, C[i,7], C[i,8] storing the intermediate vriables and C[i,6] containing the final shear force values. The shear induced by another load is added to this C[i,6] in each iteration the analysis, finally at the termination of this analysis routine this array contains the total shear force values induced by all the loads.

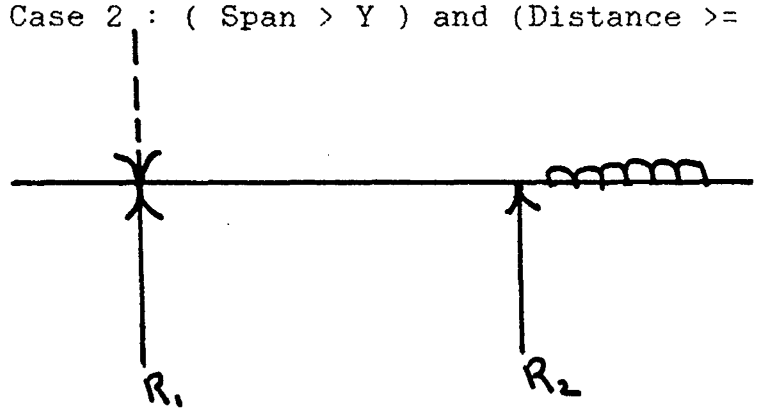
```
if (span <= y) then
  begin
     R1 := ( (span-distance) *intensity *
           (  y-distance-(span-distance)/2) )/(y-x);
     zs := (span-distance)*intensity /
           (100/(length/(span-distance)));
     zh1:=0;  g1:=g1+r1;
     for i := trunc(round(distance*100/length)) to
              trunc(round(span*100/length))     do
          begin
             c[i,7]:=zh1;
             zh1:=c[i,7]+zs;
          end;
      for B:=trunc(round(span*100/length))+1 to 100 do
          c[i,7]:=zh1;
 end
```

Here ZS is the loading on the .01 of the beam span. A FOR loop accessing from 'distance' to 'span' assigns 'ZS' to the location corresponding to the 'distance' and increment that by 'ZS' and assigns to the immediate next location and so on upto the location accessing the 'span', this simulates the loading over the span. The remaiining span is assigned the maximum value that is (Distance-Span)*intensity.

Another FOR loop from 'b' to 'd' assigns R1 to all the array location, and another FOR loop adds R2 to the array from C to D, So now C[i,8] arry corresponding from 'b' to 'c' contains the R1 and from 'c' to 'd' contains R1 + R2, this is exactly simulates the shear resitane of the supports. This values when subtracted from the shear induced by the loads gives the resultant shear force induced in which we are intrested. C[i,7] is cantaining the shear induced by the loads, So C[i,7]-C[i,8] gives the resultant shear induced.

The instruction C[i,6]:= C[i,6]+C[i,7]-C[i,8] assures the total shear induced by all the loads.

Case 2 : (Span > Y) and (Distance >= Y)



FIG

In this case the UDL is placed over the right over hanging span of the beam. This case needs a seperate routine only to compute the reactions, if reactions at both the supports are obtained the same routine above is used to compute the shear.

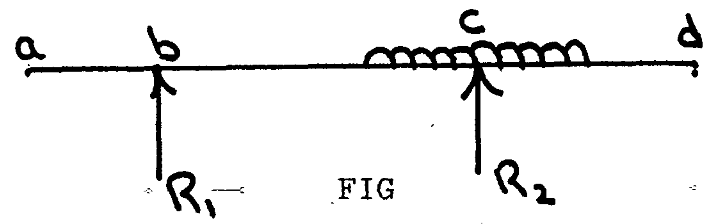
To compute the reaction, moments about the support C are cosidered

R1=((Span-distance)*intensity)-(Span-Y-(Span - Distance)/2)

R2=(Span - Distance)* intensity - R1

Case 3 : (Distance <= Y) and (Span >= Y)

In this case the udl starts from a point before the support and spans beyond the support as shown in the figure below.



FIG

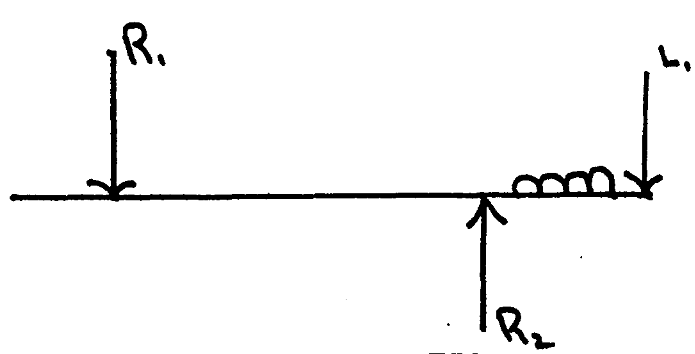
The reaction of the left support R1 and the reaction of the right support R2 are computed as below.

R1 = (Intensity*((Y-Distance)^2)/2/(Y-X) -
((Span-Y)^2)*intensity/2/(Y-X)

R2 = (Span-distance)*intensity -R1

The same routine as in Case1 is used to compute the shear forces.

At this stage, it is neccesary to check the direction of the supports assumed before are correct or not, It is assumed that the loads are acting downwards and the the supports are in upward direction, but in the case of overhanging beams if the loads act over the overhahging span the far support should be in the downwards direction, as shown below.



FIG

This can be very easyly checked by using the R1 and R2 vlaues. Since R1 and R2 are computed asssuming that the direction of the supports is upwards, and R1 and R2 must be a positive values if the direction assumed is correct, but if the R1 is negetive means the left supprt should be downwards, else if R2 is negetive R2 must act downwards.

The user is intimated about this direction changes and proceeded.

Bending moment

A single routine is developed to compute the moment of the overhanging beam, this is simple since the reaction of the supports are computed already. The moment at any point X on the beam be arrived at by taking the net of the moments acting at the point. Arrays C[i,2] and C[i,4] were used solely for this computatioins. For convienience M,N are defined, these are the integers which correspond to 'Distance' and 'Span' array locations.

```
L1 := 100   *  distance/length ;
L2 := span * 100 /length;
M  := trunc(round(l1)) ; N := trunc(round(l2));
           z:= 0;
           for i:= m to n do
              begin
                 c[i,2] := z;
                 z := (z+intensity *length/100);
              end;
           for i:= n+1 to 100 do
            c[i,2] :=intensity *(span-distance);
           for i := 0 to 100  do
            c[i,4]:= -(c[i,2]*
                      (C[I,1]/100-distance)/2 );
           for i := trunc(round(100*x/length))
                   to trunc(round(y*100/length)) do
               begin
                 if c[i,1] < span*100 then
                    c[i,4] := r1 *(c[i,1]/100-x)
                   -c[i,2]*(c[i,1]/100-distance)/ 2
                 else if c[i,1] >= span*100 then
                    c[i,4] :=  r1 *(c[i,1]/100-x)
                   -c[i,2] * (c[i,1]/100-distance
                   -(span-distance)/2 );
```

```
            end;
        for i:=trunc(round(100*y/length))to100 do
            begin
              if c[i,1] < span*100 then
                 c[i,4] := r1 *(c[i,1]/100-x)
                           +R2 * (C[i,1]/100-Y)
                           -c[i,2]*(c[i,1]/100
                           -distance)/ 2
            else if c[i,1] >= span then
                 c[i,4] := r1 *(c[i,1]/100-x)
                   + r2*(c[i,1]/100 -y)
                   -c[i,2]*(c[i,1]/100
                  -distance-(span-distance)/2);
```

Fixed Beam

For the fixed beam extra fixed end moments are computed, considering every load and finally adding to get the total moment. THe net moment is arrived at by subtracting this end moments from the moments induced by the loads, That is the advanage of using a fixed beam than a simply supported beam, because net moment induced in the fixed beam is less.

To compute the fixed end moments due to a UDL 'Area moment method' is used. According to Mohr's theorem "The area of the free and fixed bending moment are numerically equal or the resultant area of the bending moment is zero".

This leads to the equation

A' = (M1-M2)/2 + L
A'X'' = L^2(M1+2*M2)/6
where M1,M2 : moments at the ends
L length of the beam
A' area of the Fixed momemt diagram
X'' distance of C.G.

Substituting the other values and Solving both this equations the values of M1 and M2 are arrived at.

M1 is assigned to the left most point and M2 to the right most point and the intermediate values are interpolated leniarly, as

```
for i := Ø to 99 do
  begin
     z1:=z;
      z:=z+(c[i,5]+c[i+1,5])/2
          *(length/1ØØ);
  end;
uarea:=z1;
```

This routine above compues the total area of the bending moment induced by the loads, and this is assigned to 'UAREA', this correspond to A in the equatons above.

```
z:=Ø;
for i:= Ø to 99 do
   begin
     z:=z+(c[i,4]+c[i+1,4])/2*length/1ØØ *
          (c[i,1]/1ØØ+length/2ØØ);
     z1:=z;
   end;
umax:=z1;
```

The routine above computes the moment area of the bending moment diagram, this assigned to 'UMAX', corresponding to A'X''.

The fixed end moments are computed bt the routine.

```
z1 := 4*uarea/length-umax*6/length/length;
z2 := 2*uarea/length-z1;
m1 := m1+z1;
m2 := m2+z2;
```

The intermediate values are interpolated and assignd to the corresponding locations, and assigned to G[i] and the net bending moment to F[i] as

```
z  := m1;
for i:= Ø to 1ØØ do
    begin
       g[i]:=z;
       z:=z+(m2-m1)/1ØØ;
       f1:=g[i]-c[i,5];
       f[i] :=-( f[i]+f1);
    end;
```

After analysis of loads the next step is to display the analysis by a graphic routine. Infact the bending moment diagrams are better appreciated by the graphical display. This is explained clearly later.

DEAD LOAD MOMENT

Besides the shear force and bending moment induced by the loads, some amount of bending moment will be induced due to the dead load of the beam. To compute the moment induced by the dead loads, first the section of the beam has to be decided. The fial section is decided later after an interactive session between the user and the knowledge base. At the moment in the pascal environment, the dimensions of the beam are not available to compute the moment due to dead load. Since the design of the beam is based on the IS-456 codal specifications, unless user specifies his own section, even if user specifies the section, the validity and safety of the section is checked against IS-456 specifications. Since calling once again 'analysis.pas' to compute ther moment induced by the dead loads in the Prolog environment and thus the data transfer through I/O files may slow down the process of execution. So it is better to estimate approximately the dead load in the pascal environment itself strictly following IS-456 specifications.

Knowing the length of the beam, by codalspecifications, the minimum depth and breadth to be provided is computed and thus the dead load and the moment. The moments induced by tythe loads is stored in C[i,5] and the moment indiced by the derad laod is separately stored in C[i,9] and finally added and graphically displayed.

LINKING PROLOG AND PASCAl

The expert system to analyze and design beams should be capable of performing both the complex numerical computations and also the symbolic computations working with the knowledge base. It is not an easy task to implement both numerical and symbolic computations using a single programming language like Pascal or Prolog. If only Pascal is used, the knowledge and rules may be simulated and implemented but extending the program, adding few more rules is very difficult. If only prolog is used, it is easy to implement symbolic computations, but the simulation of sybolic cxomputation like arrays by lists and recursion makes the program very complicated and unreadable. The only possible solution is to use both the symbolic computation language and numerical computation language and build an effective link between them to transfer data. The linking can bve done in many ways, but here files were used to transfer data from Pascal to Prolog. The Pascal program 'analysis.pas' is called in prolog environment and executed. 'Analysis.pas' reads the input of beams and loads and analyzes them and writes the data in a text file 'data.dat' and displays graphically the analysis and terminates. Later this file is opened in Prolog environment and the data is read from 'Data.dat'. Prolog has an inbuilt predicate 'system("DOS command")', this predicate transfers control to the operating system and executes the DOS command given in the quotes and regains the control. Turbo Pascal has a provision to create an executable file, the executable file 'analysis.com' of 'analysis.pas' is created and the predicate 'system("analysis.com")' executes the file 'analysis.com', which analyzes the beam and creates the file 'data.dat' in Prolog environment.

The disadvantage of this technique of transfering data through I/O files is relatively slow compared to transfer of data by linking of routines, but still the data transfer through files is prefered due to it's simplicity and flexibility. Moreover the data transfer is performed only once. The data read from the 'data.dat' is stored in the knowledge base which can be referred at any stage of execution.

The grade of cocrete, the grade of steel and the diameter of the bars, the type of beam and loading specifications were read by interaction in 'analysis.pas' and the loads are analyzed and all this data is written in data.dat. The grade of the concrete and the grade of steel specified by the user are checked for the safe design. If they are found to be unsafe, and changes are needed. This is intimated to the user and requested to change the specifictions. If still the user insists in using the same grades, other alternatives are considered. A small routine is used to pick up the maximum and minimum shear force and bending moment values and the moments at different points were picked up and along with other data were written in 'data.dat'.

GRAPHICAL DISPLAY

The analysis is better appreciated by the user if the output is displayed graphically. Obviously the variations of the moment and shear should be displayed graphically rather than numerically. The graphics routine is called if the usser wants the output graphically. If the beam loaded with heavy loads, the moment values are high and low if loaded with relatively low intensity of loads. The bending moment diagram should not be so small if the loads are of low magnitude beyond the comprehension of the human eye and should not be very large abnormally also. So a relative diagram is displayed. The maximum value of bending moment is picked and 40 pixels were alloted to represent that vlaue and all orther values are displayed relatively. So a beam loaded with UDL it's full length with 1.0.T/m, 10.T/m, 100.T/m will display exaxctly the same bending moment diagram. The actual implementation is

```
begin
  graphcolormode;
  r:=0;
  for i:= 0 to 100 do
    if abs( c[i,5]) > r then r:=abs( c[i,5]);
      if r<>0 then
        begin
          j:=48;
          draw (50,50,248,50,1);
          for i:= 1 to 100  do
            begin
               l:= trunc(round(c[i,5]/ r * 40.0));
               j:=j+2;
               k:=50-l  ;
               draw(j,50,j,k,2)
            end;
```

In case of fixed beam, first the moment diagram is displayed and over it, the separately computed fixed end moments are super imposed and finally below it, the resultant of both the moments is dispalyed. A single routine is used to display the bm diagram for all the four types of beams.

```
    if fxflag=true then
       begin
         j:=49;
         draw (50,90,250,90,1);
         for i:= 0 to 100  do
            begin
              l:= trunc(round(g[i]/ r * 40.0));
              j:=j+2;
              k:=50-l  ;
              draw(j,50,j,k,9);
            end;
         j:=49;
           for i:= 0 to 100 do
              begin
                 begin
                   l:= trunc(round(f[i]/ r * 40.0));
                   j:=j+2;
                   k:=90-l  ;
                   draw(j,90,j,k,2);
                 end;
                 r:=0;
           for i:= 1 to 100 do
             if abs(c[i,6]) > r then r:=abs( c[i,6]);
               if r<>0 then
                 begin
                    j:=48;
                    draw (50,160,248,160,1);
                    for i:= 1 to 100  do
                      begin
                        l:= trunc(round(c[i,6]/ r * 40.0));
                        j:=j+2;
                        k:=160-l  ;
                        draw(j,160,j,k,2)
                       end;
                 end;
```

DESIGN IMPLEMENTATION IN PROLOG

The design part of the program has been implemented om Prolog. This section clearly explains the techniques used for the implementation. Prolog programs are basically a declarative. The decleration were made in the 'Domains',' Predicates', 'Clauses'.

Domains : "mf" is declared as a file and later assigned to data.dat and opened to read.

Data base : data base contains the facts which are computed and generated during the process of execution.The computed values which are needed many times during the execution are also stored as facts. This technique of storing the variables as facts may increase the 'search' in the program. But transfering many variables from one predicate to the other makes the program look complicated. In the progarms which are relatively small, it is better to store the variables in the knowledge base and later extracted. The beam(string), steel(integer), stldia(real), length(real), pbmd(real).The length of the beam is extracted from length(X),X is bounded to the real value of length(real).The safe breadth and depth computed as per IS-456 are stored as safe_breadth(real),safe_depth(real) and the final breadth and depth which is decided later by interaction as a final_breadth(real) and final_depth(real).

Predicates : The predicate section is declared with numerous predicates each contributing a solution for the small divided problem of the major task of designing the beam. The flow of control and data is described clearly in the flow-chart.

Start : The predicate start is the first and at the top of the hierarchy. In fact, the goal of the program is to prove the predicate 'start' true. This predicate triggers all other predicates. This predicate tries to prove 'True' all the predicates.

```
start :- write("Letsstart"), openread(mf,"data.dat"),
readdevice(mf),readln(Beam), readreal(LENGTH), readln(CC),
readint(SS), readreal(NB), Nbmd=1.5*Nb, readreal(PB),
Pbmd=Pb*1.5, readreal(Sh1), S1=Sh1*1.5, readreal(Sh2),
S2=Sh2*1.5, readreal(Stldia), readreal(Depth),
readreal(Breadth), closefile(mf), asserta(beam(Beam)),
asserta(concrete(CC)),asserta(steel(SS)),asserta(nshr(Sh1)),
asserta( pshr ( Sh2 ) ), asserta( length ( LENGTH ) ),
asserta( nbmd ( NBMD ) ), asserta( pbmd ( PBMD ) ),
asserta( pascald( Depth) ), asserta ( pascalb( Breadth )),
asserta(stldia(Stldia)), window, bmfactor, !, conc, !, stl,
cvr(Stldia), safe_section(Length), option, breadth, design.
```

This predicate first tries to prove true a write statement and prints the message of introduction. The next predicate is 'system("analysis.com")'. This is a very important predicate as explained before which links Pascal and Prolog. By executing the single predicate, all the task of reading the input and loads analyzing the loads and displaying it graphically and creating a file of data. The next task is to open the data file 'data.dat' created by 'analysis.com' and read the data and store them in the data base.

The values witten in the file were read and stored in the knowledge base by using the predicate "asserta(length(Length))", this predicate asserts the characters in the quotes as a clause to the program. The shear force and bending moment values were multiplied by 1.5 the partial safety factor to correspond to the limit state of design. The next predicate creates a window with the header 'Auto beam" in reverse video. All the interactions and input out put is through this window. The predicate 'bmfactor' decides a factor which is the maximum ratio of length and depth of the beam section. This gives the minimum depth to be provided for the safety against the deflection of the beam under loading. IS-456 clearly specfies the minimum ratios of length and depth to be provided for safety. This factor is computed by this predicate and assertas in the database 'bmfac(Factor)'. The next predicate to be satisfied is 'conc', this predicate asserts ther flexural strength of concrete given the grade of concrete, and the precdicate 'stl', asserts 'stlgrd(Grade)' and also asserts a variable xlim(X) which is derived by the grade of steel and needed later for computations. The 'cover' predicate decides the minimum cover to be provide depending on the diameter of the bars to be used for reinforcement. This diameter is specifed by the user, if not it is arrived at by knowing the length and the depth is estimated, from which the diameter is assumed. The next predicate triggered is the 'safe_section' predicate, this decdes the safe section dimensions for the beam as per IS-456. The predicate 'option' writes the message that the expert system designs a singly reinforced section if the user does not specify the dimensions of the beam. This is necessary because the user may specify the loading, the grade of steel and concrete he likes to use and lever other details to the system itself then the user must be informed that the beam going to be designed is a singly reinforced but not doubly reinforced.

Breadth : The predicate breadth decides the section of the beam. There can be infinite sections which can safely handle the loads and moments thereby induced, but only one section can be opted for. The program displays six possible sections which can safely handle the loads with different depth breadth ratios varying from 1.5 to 4. This acts as a guidance to the user if he wants to specify the section. Next the user is questioned if he wants to specify depth/ breadth or both. If the user specifies the depth,the breadth is decided else if breadth specified, depth is decided and if both the dimensions were specified by the user, the validity and safety of the section is checked. If the section specified is proved unsafe, other changes in the design are made. This predicates breadth displays this information and calls decide_depth predicate.

The decide_depth predicate computes the sections with depth/breadth ratios varying from 1.5 to 4 and displays them and triggers 'dcd'. The 'dcd' predicate reads the response of ther user and calls 'dcd1(Reply)'.

dcd1(N): A message is prited that the design is based on IS-456 specifications and a single reinforcement beam is going to be designed and triggers 'prélim_design'.

dcd1(Y): This predicate reads the breadth or depth the user wants to specify and calls decide_b(Depth) if depth is specified or decide_d(Breadth) if breadth is specified.

decide_b(D) : This predicate has to decide the breadth. The predicates length(L),x_lim(X) etc. when tried to prove true, prolog searches the knowledge base and binds 'L' and 'X'. This is how we can regain the variable values from the knowledge base. The breadth is arrived by substituting other values in the equation.

$$M = 0.36 * Fck * B * D * D (1 - 0.42 * X) * X$$

where B and D are breadth and depth of the beam. The breadth computed by this equation is rounded and a clear cut data of the situation is presented to the user by printing the values of the design moment 'M', depth specified by the user 'D', bredth which can safely handle the loads 'B' and the minimum breadth to be provided as per IS-456. The greater of both the breadths is selected for design and the user is questioned " Is the section satisfactory?". The response of the user is bound to the Reply and 'hm1(Reply)' is triggered.

decide_d(B) : This predicate decides the depth of the beam section io the user specifies the breadth and depth to be decide by the system. This predcate as 'decide_b' not only computes the other dimension but also gives the minimum depth should be povided by considering the modification factors into account. The depth provisions in IS-456 can be further modified considering the modification factors which makes the design more economical. The predicate 'modfac' is called ,which approximately computes the percentage of steel in the section and reads the corresponding value for the grade of steel from the data base as below.

```
modfac_tsn(Ø.Ø,2.1ØØ,2.4ØØ,3.ØØØ).
modfac_tsn(Ø.2,1.35Ø,1.6ØØ,2.3ØØ).
modfac_tsn(Ø.4,1.Ø75,1.28Ø,2.1ØØ).
modfac_tsn(Ø.6,Ø.95Ø,1.1ØØ,1.73Ø).
modfac_tsn(Ø.8,Ø.9ØØ,1.Ø25,1.525).
modfac_tsn(1.Ø,Ø.85Ø,Ø.95Ø,1.415).
modfac_tsn(1.2,Ø.825,Ø.925,1.32Ø).
modfac_tsn(1.4,Ø.81Ø,Ø.9ØØ,1.285).
modfac_tsn(1.6,Ø.8ØØ,Ø.88Ø,1.215).
modfac_tsn(1.8,Ø.79Ø,Ø.86Ø,1.15Ø).
modfac_tsn(2.Ø,Ø.775,Ø.84Ø,1.125).
modfac_tsn(2.2,Ø.76Ø,Ø.82Ø,1.1ØØ).
modfac_tsn(2.4,Ø.745,Ø.8ØØ,1.Ø85).
modfac_tsn(2.6,Ø.73Ø,Ø.785,1.Ø65).
modfac_tsn(2.8,Ø.715,Ø.77Ø,1.Ø45).
modfac_tsn(3.Ø,Ø.7ØØ,Ø.755,1.Ø25).
```

Knowing the grade of steel and the percentage of steel the corresponding ratio is computed by interpolation of the two nearest values. After the display of the section if the user in satisfied the predicate gm1,gm2 are called as explained in the predicate 'decide_b'.

hm1(Reply) : If the Reply is 'y', the user is thanked for considering the advice of the system and proceeded to design. Since the breadth and depth are decided. The reply is 'N', the message to specify the breadth or depth have to be changed. The response is bound to A and hm2(A) is called.

hm2(A) : If A is equal to 'D', then the user is requested to enter the depth he prefers which is bound to 'D'. At this stage, the user wants to change the depth he specified first. So once again the predicate 'decide_b(D)' is called else if A is 'B', the breadth is read and bound to 'B'. At this stage, the user first specified the depth and later specified the breadth considering the data displayed. So the expert system is provided with both the dimensions. The next step is to chech the validity of that section. Perilim_design(B,D) is called.

prelim_design(B,D) : This predicate decides if the breaths and depth are capable of handling the loads specified by a singlly reinforced section. If not safe, the user is questioned if the design can be for a doubly reinforced beam. iF the rersponse is 'Y', 'double(B,D)' is called.If the response is 'N' the predicate, means the user does not want the beam to be design a doubly reinforced. So the program is backtracked to the predicate 'breadth', once agin teh user is facilitated to chage the the dimensions of the section.

The user after deciding the section, and the sectioon is proved safe satisfying all the codal specifications the predicate 'design' is triggered.This predicate designs the final design and the percentage of steel and the steel details are fixed y the predecate 'stl_cal'. The knowledge base is furnished with the safe and minimum percentages to be satisfied as below.

```
max_prsnt_rnf_sngl(15,250,1.32).
max_prsnt_rnf_sngl(15,415,0.72).
```

```
max_prsnt_rnf_sngl(15,500,0.57).
max_prsnt_rnf_sngl(20,250,1.76).
max_prsnt_rnf_sngl(20,415,0.96).
max_prsnt_rnf_sngl(20,500,0.76).

max_prsnt_rnf_sngl(25,250,2.20).
max_prsnt_rnf_sngl(25,415,1.19).
max_prsnt_rnf_sngl(25,500,0.94).

max_prsnt_rnf_sngl(30,250,2.64).
max_prsnt_rnf_sngl(30,415,1.43).
max_prsnt_rnf_sngl(30,500,1.13).
```

This data is asper IS-456 and gives the extreme values of the reinforcement.

```
PROGRAM Anlysis(input,output);
 VAR
     c                                                          :packed
                                          array[0..100,1..10] of real;
     g,f,h                              packed array[0..10]    of real;
     l1,l2,s1,s2,g1,sh1,sh2,g2,g3,g4,zs,zh1,i1,i2       :real;
     r1,r2,x,y,p,z,a1,a2,r,m1,m2,z1,z2,f1,e1            :real;
     i,j,m,n,l,k,grade                                   :integer;
     answere,bmtype                                      :string[2];
     concgrd,reply                                       :string[3];
     over,entry,flag,ssflag,ohflag,clflag                :boolean;
     fxflag,ptflag,udlflag,sdflag                        :boolean;
     Rint,Lint,span,intensity,length,distance            :real;
     parea,uarea,pmax,umax,smx,dcudl,rat,modfac          :real;
     breadth,depth,stldia                                :real;
     fil1,fil2                                           :text;

     procedure mdf;
      var
        A                    :packed array[1..16,1..4] of real;
        z1,z2,p              :real;
        i,z3                 :integer;
       begin
         p:=0;
         for i:= 0 to 16 do
             begin
                a[i,1] := p;
                p      := p+0.2;
             end;

         a[1,2]  :=2.100; a[1,3]  :=2.400; a[1,4]:=3.000;
         a[2,2]  :=1.350; a[2,3]  :=1.600; a[2,4]:=2.300;
         a[3,2]  :=1.075; a[3,3]  :=1.280; a[3,4]:=2.100;
         a[4,2]  :=0.950; a[4,3]  :=1.100; a[4,4]:=1.730;

         a[5,2]  :=0.900; a[5,3]  :=1.025; a[5,4]:=1.525;
         a[6,2]  :=0.850; a[6,3]  :=0.950; a[6,4]:=1.415;
         a[7,2]  :=0.825; a[7,3]  :=0.925; a[7,4]:=1.320;
         a[8,2]  :=0.810; a[8,3]  :=0.900; a[8,4]:=1.285;

         a[9,2]  :=0.800; a[9,3]  :=0.880; a[9,4] :=1.215;
         a[10,2] :=0.790; a[10,3] :=0.860; a[10,4]:=1.150;
         a[11,2] :=0.775; a[11,3] :=0.840; a[11,4]:=1.125;
         a[12,2] :=0.760; a[12,3] :=0.820; a[12,4]:=1.100;

         a[13,2] :=0.745; a[13,3] :=0.800; a[13,4]:=1.085;
         a[14,2] :=0.730; a[14,3] :=0.785; a[14,4]:=1.065;
         a[15,2] :=0.715; a[15,3] :=0.770; a[15,4]:=1.045;
         a[16,2] :=0.700; a[16,3] :=0.755; a[16,4]:=1.025;

         z3 := trunc(rat/0.2);
         z1:= a[z3+1,grade+1];
         z2:= a[z3+2,grade+1];
         modfac:= z1+ (z1-z2) * (rat-a[z3+1,1])/0.2;
         writeln(' modigficatioon factor : ',modfac);
```

```
   end;

    procedure cal;

     begin
          p:= 0;
          for i:= 0 to 100 do
              begin
                   c[i,1] := p;
                   p:=p+length;
              end;
          p:=0;
          for i:= 0 to 100 do
             begin
                  c[i,2] := p;
                  p       := ddudl*c[i,1]*c[i,1]/20000.0;
             end;
          z:=ddudl*length/2;
          p:=0;
          for i:= 0 to 100 do
            begin
               c[i,9] := (z*c[i,1]/100) - (c[i,2]);
            end;
      end;
begin
     clrscr;
     over := true;
     while over=true do

        begin
          over := false;
          for I:= 0 to 100 do
             begin
                c[i,1]:=0;
                c[i,2]:=0;
                c[i,3]:=0;
                c[i,4]:=0;
                c[i,5]:=0;
             end;
          clflag:=false;ssflag:= false;
          ohflag:= false;fxflag:=false;
          ptflag:=false;udlflag:=false;

          flag := true;
          writeln;
          writeln ('Which type of beam you like to design');

          while flag= true do
              begin
                 flag:= false;
                 writeln;
                 writeln('The following options are avilable');
                 writeln('Cantilever beam          [cl]');
                 writeln('Simply supported beam   [ss]');
                 writeln('Over hanging beam       [oh]');
```

```
    writeln('Fixed beam                [fx]');
    writeln;
    write('Enter your selected  option by entering the');
    write(' characters in the braces : ');
    readln (answere);bmtype    :=answere;
    if (( answere ='ss') or (answere ='SS'))
      then ssflag:= true
      else if((answere='oh')or(answere='OH'))
           then ohflag:= true
             else if ((answere='fx')or(answere='FX'))
                    then fxflag := true
                else if (answere='cl') or (answere='CL')
                        then clflag:=true
                   else
                      begin
                        write('       Invalid option ');
                        write('please try again');
                        writeln;
                        flag:=true;
                      end;

   end;
flag:= true;
     while flag = true do
       begin
         flag := false;
         clrscr;
         write(' Which grade of steel');
         wtiteln(' is used for reinforcement');
         writeln;
         writeln('The following  options are available');
         writeln('Type   " 1 "  for  FE 500  grade steel);
         write('Type  " 2 "  for  FE 415  grade steel --');
         write(' Ribbed Tar');
         writeln('Type   " 3 "  for  FE 250  grade steel');
         writeln;
         write( 'What is your option :' );read(i);
         if ( i= 1)  or ( i = 2) or ( i = 3)
           then grade:=i
           else
             begin
               writeln('  Sorry invalid option try again');
               writeln;
               flag := true;
             end;
         writeln;
         write('Specify the diameter of the steel');
         write(' bars[y/n] : ');readln(answere);
         if (answere='y') or (answere='Y') then
          begin
            flag:=true;
            while flag=true
              begin
                flag:=false;
                write('Enter the diameter of bars :');
```

```
                readln(stldia);
                if stldia=0 then flag:=true;
              end;
           end;
        if (answere='n') or (answere='N')  then
             begin
                sdflag:=true;
                stldia:=1.8;
             end;
       end;
flag:= true;
     while flag = true do
       begin
         flag := false;
         clrscr;
         writeln;
         writeln('Which grade of concrete you prefer   ');
         writeln;
         WRITELN('The following  options are available');
         writeln('M10 ,M15 ,M20 ,M25 ,M30 ,M35 ,M40 ' ) ;
         writeln;
         write('Enter your selected option: ');
         readln(reply);
         if (reply = 'm10')  or  (reply = 'M10')  or
            (reply = 'm15')  or  (reply = 'M15')  or
            (reply = 'm20')  or  (reply = 'M20')  or
            (reply = 'm25')  or  (reply = 'M25')  or
            (reply = 'm30')  or  (reply = 'M30')  or
            (reply = 'm35')  or  (reply = 'M35')  or
            (reply = 'm40')  or  (reply = 'M40')
           then concgrd:=reply
           else
             begin
               writeln('  Sorry invalid option try again');
               writeln;
               flag := true;
             end;
       end;
clrscr;
writeln;
writeln('The suppports are assumed to act ');
writeln ('in the upward direction');;
writeln('and the  loads in the downward direction');
writeln  ;
write('Please enter the length of the beam (mts) :');
readln(length);
if (bmtype ='cl') or (bmtype='Cl') then depth := length/7 ;
if (bmtype ='ss') or (bmtype='SS') then depth := length/20;
if (bmtype ='oh') or (bmtype='OH') then depth := length/26;
if (bmtype ='fx') or (bmtype='FX') then depth := length/26;

{ if depth-trunc(depth)>0.5 then depth := trunc(depth)+0.5;
 if depth-trunc(depth)<0.5 then depth := trunc(depth);}
breadth:=depth/2;
writeln('The expert opinion is to have ');
```

```
writeln('Maximum depth [mts] : ', depth);
writeln('Maximum breadth[mts]: ',breadth);
write('This approximate depth and breadth satisfactory : ');
readln(answere);
if (answere='n') or (answere='N') then
begin
  write('     Do you have any commitments regarding');
  write(' the breadth or depth of the  beam[y/n]');
  readln(answere);
  if (answere ='y') or (answere='Y') then
   begin
     write('Breadth or Depth [b/d]');
     readln(reply);
     if (reply = 'b') or (reply='B') then
       begin
          write('Enter the breadth of the beam specified: ');
          readln(breadth);
       end;
     if (reply = 'd') or (reply='D') then
       begin
          write('Enter the depth of the beam specified: ');
          readln(depth);
       end;
   end;
end;
ddudl:=breadth*depth*2.5;

cal;{ calling the procedure to calculate dead load moment }

writeln;
writeln;
if clflag= true then
      begin
         x:=0;
         y:= length;
         writeln('Now please enter the loads');
         writeln;
      end;
if ssflag= true then
      begin
         x:=0;
         y:= length;
         writeln('Now please enter the loads');
         writeln;
      end;
if ohflag= true then
         begin
           flag := true;
           while flag = true do
                begin
                  flag:=false;
                  write('Please enter the left support');
                  write('  of the beam  from left(mts) : ');
                  readln(x);
                  if (x>length) or (x<0)  then
```

```
                    begin
                      writeln;
                      write('Left support beyond span');
                      write(' please check ');
                      flag:=true;
                    end;
                end;
           writeln ;flag := true;
           while flag = true do
               begin
                  flag:=false;
                  write('Please enter right support(mts):');
                  readln(y);
                  if (y>length) or  (y<x)  then
                     begin
                        if y> length then  writeln(' Right support ');
                        write('exceedsthe beam span please check ');
                        if y<x  then writeln(' Right support is');
                        write('left of left support please check');
                        write(' and try again ');
                        flag:=true;
                     end;
               end;
               p:= 0;
               for i:= 0 to 100 do
                  begin
                    c[i,1] := p;
                    p:=p+length;
                  end;
           writeln('Please enter the the loads. ');
         end;
if fxflag= true then
   begin
       x:=0;
       y:= length;
       writeln;
       writeln('Please enter the the loads from the left.');
       write(' Any sinking of any the two supports[y/n] :');
              readln(answere);
              if ((answere='y')or(answere='Y'))then
                begin
                  writeln ('Give the values
                     of E & I unit kg/m');
                  write('E:');readln(E1);
                  write('I:');readln(I1);
                  flag:=true;
                  while flag=true do
                   begin
                      flag:=false;
                      write('reply[L/R]:');
                      readln(answere);
                      if (answere='r') or (answere='R')
                          or (answere= 'L')or
                             (answere='l')then
                             write('  ')
```

```
                        else
                           begin
                             write('Sorry try again');
                             writeln;
                             flag := true;
                           end;
                    end;
                  write('Enter the sinking in Cms :');
                  readln(snk);
                  s1 :=6*e1*I1*snk/length/length;
                  s2 := s1/50;
                  z:=s1;
                  for i:= 1 to 100 do
                     begin
                       h[i]:=z;
                       z:=z-s2;
                     end;
                  if (answere='r') or (answere='R' )then
                    for i:= 0 to 100 do h[i]:=-h[i];
                  for i:= 0 to 100 do writeln(h[i]);
                end;
  end;

 p:= 0;
 for i:= 0 to 100 do
     begin
          c[i,1] := p;
          p:=p+length;
     end;
entry :=true;
parea:=0;uarea:=0;m1:=0;m2:=0;f1:=0;z:=0; g1:=0;g2:=0;
for i:= 0 to 100 do
  begin
     c[i,6] :=0;
     c[i,7] :=0;
     c[i,8] :=0;
  end;

while entry =true do
   begin
     for i:= 0 to 100 do begin
                          c[i,2] :=0;
                          c[i,3] :=0;
                          c[i,4] :=0;
                          f[i]   :=0;
                          g[i]   :=0;
                        end;
     entry := false;
     flag:= true;

     while flag = true do
       begin
         flag := false;
         writeln;
         WRITELN('The following load options are available');
```

```
    writeln('Type    " 1 "   for concentrated load' ) ;
    writeln('Type    " 2 "   for distributed load ');
    writeln;
    write( 'What is your option :' );readln(answere);
    ;
    if (answere = '1')  or (answere =  '2')
      then writeln
      else
        begin
          writeln;
          writeln('  Sorry invalid option try again');
          flag := true;
        end;

  end;
if answere ='2' then  udlflag := true;
if answere = '1' then
  begin
    flag := true;
    ptflag:=true;
    while flag =true do
      begin
        write('Enter the load from left(mts) :');
        readln(distance);
        writeln;
        flag:= false;
        if distance > length then
           begin
            writeln('Load exceeds the span try again');
             flag:= true;
           end;
      end;
    write('Magnitude of the load :');
    readln(intensity);
    writeln;
    g1 := g1 + intensity*(y-distance);
    g2 := g2 + intensity;

    if clflag=true then
      begin
          z:=0; z1:=0;
          for i:=trunc(round(distance*100/length))
               downto 0 do
            begin
               c[i,9]:=c[i,9]+z;
               z:=z+intensity*length/100;
               c[i,10]:=c[i,10]+intensity;
            end;
          for i:= 0 to 100 do
            begin
               c[i,5]:=c[i,9];
               c[i,6]:=c[i,10];
           end;
      end;
    if (distance > x ) and (distance < y ) then
```

```
    begin
      for i:= trunc(round(distance*100/length))+1
            to 100 do
              c[i,8] := intensity;

      for i:= 0 to 100 do
        begin
           c[i,7] := c[i,7] +c[i,8];
           c[i,8]:=0;
        end;
      lint := (intensity/100 *(distance-x)*
              (y-distance)/(y-x))
              /((distance-x)/length);
      z:=0;
      for i:=trunc(round(100 *x/length))   to
             trunc(round(distance /length*100))  do
        begin
          c[i,2]:=z;
          z:=z+lint;
        end;
      Rint := (intensity/100 *(distance -x)   *
              (y-distance)/(y-x))/
              ((y-distance)/length);
      z:=0;
      for I:= trunc(round(100 * y/length))   downto
              trunc(round(distance /length*100))do
        begin
          c[i,2] := z   ;
          z :=z + rint  ;
        end;
    end;
 if (distance < x ) or (distance > y ) then
   begin
     if distance < x then
       begin
          for i:=trunc(round(distance*100/length))+1
                    to 100   do
              c[i,8] := intensity;
          for i:= 0 to 100 do
              begin
                c[i,7] := c[i,7] +c[i,8];
                c[i,8]:=0;
              end;

         z:=0;
         lint := intensity/100*(x-distance) /
                 ((x-distance)/length);
         for I:= trunc(round(distance /length*100))
              to trunc(round(100* x/length)) do
                  begin
                    c[i,2] := -z;
                    z:=z+lint;
                  end;
         writeln(z);
         lint := intensity/100*(x-distance)/
```

```
                ((y-x)/length);
        z:=0;
        for I:=trunc(round(100*y/length))  downto
                trunc(round(100* x/length)) do
          begin
            c[i,2]:=-z;
            z:=z+lint;
          end;
      end;
    if distance > y then
      begin
         for i:=trunc(round(distance*100/length))+1
                 to 100 do
             c[i,8] := intensity;
         for i:= 0 to 100 do
             begin
               c[i,7] := c[i,7] +c[i,8];
               c[i,8]:=0;
             end;

        rint := intensity/100*(distance-y) /
                ((distance-y)/length);
        z :=0;
        for I:= trunc(round(100* distance/length))
            downto trunc(round(y /length*100)) do
              begin
                c[i,2] := -z;
                z:=z+rint;
              end;
        z:=0;
        rint :=  intensity/100*(distance-y)/
                   ((y-x)/length);
        for I:= trunc(round(100*x/length))  to
                trunc(round(y /length*100)) do
         begin
           c[i,2] := -z;
           z:=z+rint;
         end;
      end;
   end;
  if clflag=false then
   begin
       for I:= 0 to 100 do
          begin
             c[i,4] := c[i,4] +c[i,2];
             c[i,5] := c[i,5] +c[i,4];
          end;
   end;
if fxflag=true then
  begin
    if ptflag=true then
      begin
         for i:=trunc(round(distance*100/length))+1
                 to  100 do
             c[i,8] := intensity;
```

```
              for i:= 0 to 100 do
                  begin
                    c[i,7] := c[i,7] +c[i,8];
                    c[i,8]:=0;
                  end;

            z1:=intensity*distance * (length-distance) *
                   (length-distance)/(length*length);
            z2:=intensity * distance * distance *
                   (length-distance) / (length*length);
            m1:=m1+z1;
            m2:=m2+z2;
            z:=m1;
            a1:=(m2-m1)/100;
              for i:= 0 to 100 do
                  begin
                     g[i]:=z;
                     z:=z+a1;
                     f1:=g[i]-c[i,5];
                     f[i] :=-( f[i]+f1);
                  end;
           end;
       end;
     write('Any other loads ?[y/n] :');
     readln(answere);;writeln;
    if ((answere='y')or(answere='Y')) then entry := true
        else
         if clflag <>true then
           begin
               entry:=false;
               g3 :=  g1/(y-x);
               g4 := g2 -g3 ;
               for i:= trunc(round(x*100/length))+1
                        to 100 do
                  c[i,8] := g3;
               for i:= trunc(round(y*100/length))+1
                        to 100 do
                  c[i,8] :=c[i,8]+g4;
                   sh1:=0;sh2:=0;
               for i:= 0 to 100 do
                    begin
                       c[i,6] :=c[i,6]+c[i,8]-c[i,7];
                       if c[i,6]>sh2 then sh2:=c[i,6];
                       if c[i,6]<sh1 then sh1:=c[i,6];
                    end;
           end;
   end;
-----------------------------------------------------------------*)

  while udlflag =true do
    begin
      flag := true;
      while flag =true do
        begin
          flag := false;
```

```
      writeln;
      write('    Please give the point from left ');
      write('  where the udl starts(meters) :');
      readln(distance);
      if distance <0 then
         begin
           writeln(' loading beyond the limits of');
           write(' beam check and try again');
           flag := true;
         end;
    end;
  flag:=true;
  while flag =true do
    begin
      flag := false;
      writeln;
      writeln;
      write('     please give where the udl ends');
      write('from left(meters) : ');
      readln(span);
      if span >length then
         begin
           write('The udl  spans beyond the beam');
           write(' please check and try again ');
           flag := true;
         end;
    end;
  writeln;
  write('    Please give the intensity of the');
  write(' udl(tons/meter) : ');
  readln(intensity);
  if clflag=true then
    begin
        l1:=distance;
        l2:=length/100;
        z:=(span-distance)*intensity*
            (span-distance)/2;
        z1:=intensity*length/100;
        s1:=(span-distance)*intensity;
        s2:=span;
        for i:= trunc(round(distance*100/length)) to
                trunc(round(span*100/length)) do
           begin
              c[i,9]  :=c[i,9]+z;
              z:=(s2-distance)*intensity
                  *(s2-distance)/2;
              s2:=s2-l2;
              c[i,10]:=c[i,10]+s1;
              s1      :=s1-z1;
           end;
        z:=0;   s1:=(span-distance)*intensity;
        for i:=trunc(round(distance*100/length))-1
                downto 0 do
            begin
              z:= s1*(span-l1)/2;
```

```
            c[i,9]  :=c[i,9]+z;
            l1:=l1-l2;
            c[i,10]:=c[i,10]+s1;
         end;

      for i:= 0 to 100 do
        begin
            c[i,5]:=c[i,9];
            c[i,6]:=c[i,10];
       end;
  end;
 if (span <= y) then
  begin
    R1 := ( (span-distance) *intensity *
          ( y-distance-(span-distance)/2))/(y-x);
    zs:=(span-distance)*intensity /
        (100/(length/(span-distance)));
    zh1:=0;
    g1:=g1+r1;
    for i:=trunc(round(distance*100/length)) to
           trunc(round(span*100/length))      do
      begin
         c[i,7]:=zh1;
         zh1:=c[i,7]+zs;
      end;
    for i:=trunc(round(span*100/length))+1
         to 100 do
        c[i,7]:=zh1;
  end
else
  begin
    if distance>= y then
      begin
         R1:=-((span-distance)*(span -distance)/2)*
              intensity;
         zs:=(span-distance)*intensity /
             (100/(length/(span-distance)));
         zh1:=0;
         g1:=g1+r1;
         for i:=trunc(round(distance*100/length))
             to trunc(round(span*100/length)) do
           begin
             c[i,7]:=zh1;
             zh1:=c[i,7]+zs;
           end ;
         for i:=trunc(round(span*100/length))
                to 100 do
         c[i,7]:=zh1;
      end;
    if distance<y then
      begin
          R1:= (( intensity * (y-distance)*
                (y-distance) /2)/
                (y-x))-((span-y)*(span- y)*
                 intensity /2 /(y-x));
```

```
          zs:=(span-distance)*intensity /
              (100/(length/(span-distance)));
          zh1:=0;
          g1:=g1+r1;
          for i:=trunc(round(distance*100/length))
              to trunc(round(span*100/length))  do
              begin
                c[i,7]:=zh1;
                zh1:=c[i,7]+zs;
              end  ;
          for i:=trunc(round(span*100/length))
                  to 100 do
              c[i,7]:=zh1;
      end;
  end;
R2 := (span - distance )*intensity - R1;
if (r1 < 0)   or  ( r2 < 0) then
  begin
    if r1<0 then  write(' The left support ');
          write('should in downward direction');
    if R2<0 then  write(' The right support ');
         write('should downward direction');
    write(' Don't worry  it is assumed that the');
    write(' support is in opposite direction ');
  end; z:=0;
L1 := 100 * distance/length ;
L2 := span * 100 /length;
m :=trunc(round(l1)) ;n := trunc(round(l2));
z:= 0;
for i:= m to n do
   begin
      c[i,2] := z;
      z := (z+intensity *length/100);
   end;
for i:= n+1 to 100 do
 c[i,2] :=intensity *(span-distance);

for i := 0 to 100  do
 c[i,4] := - (c[i,2]* (C[I,1]/100-distance)/2 );
for i  := trunc(round(100*x/length))
         to trunc(round(y*100/length)) do
            begin
              if c[i,1]  < span*100 then
                c[i,4]  := r1 *(c[i,1]/100-x)
              - c[i,2]*( c[i,1]/100-distance)/ 2
              else if c[i,1] >= span*100 then
                c[i,4]  :=  r1 *(c[i,1]/100- x)
                           - c[i,2] *
     ( c[i,1]/100-distance-(span-distance)/2 );
            end;

for i  := trunc(round(100*y/length)) to 100 do
          begin
            if c[i,1]  < span*100 then
                c[i,4]  := r1 *(c[i,1]/100-x)
```

```
                +R2 * (C[i,1]/100-Y)
              - c[i,2]*(c[i,1]/100-distance)/ 2
           else if c[i,1] >= span then
               c[i,4] := r1 *(c[i,1]/100-x)
                  + r2*(c[i,1]/100 -y)
                  - c[i,2]*( c[i,1]/100-distance
                  -(span-distance)/2 );
         end;
if c1flag=false then
for i := 0 to 100 do c[i,5]:=c[i,5]+ c[i,4];
writeln;
if fxflag=true then
  begin
    if udlflag=true then
      begin
         zs:=(span-distance)*intensity /
             (100/(length/(span-distance)));
         zh1:=0;
        for i:=trunc(round(distance*100/length)) to
              trunc(round(span*100/length))      do
          begin
            c[i,7]:=zh1;
            zh1:=c[i,7]+zs;
          end;
   for i:=trunc(round(span*100/length)) to 100 do
            c[i,7]:=zh1;
         z:=0;
         for i := 0 to 99 do
             begin
                z1:=z;
                z:=z+(c[i,5]+c[i+1,5])/2
                    * (length/100);
             end;
         uarea:=z1;
         z:=0;
         for i:= 0 to 99 do
            begin
              z:=z+(c[i,4]+c[i+1,4])/2*length/100 *
                   (c[i,1]/100+length/200);
              z1:=z;
            end;
         umax:=z1;
         z1 := 4*uarea/length-umax*6/length/length;
         z2:= 2*uarea/length-z1;
         m1:=m1+z1;
         m2:=m2+z2;
         z:=m1;
         for i:= 0 to 100 do
             begin
                g[i]:=z;
                z:=z+(m2-m1)/100;
                f1:=g[i]-c[i,5];
                f[i] :=-( f[i]+f1);
             end;
      end;
```

```
                end;

              write('    Any other loads ?[y/n]: ');
              udlflag:=false;
              readln(answere);;
              if ((answere='y')OR(answere='Y'))then entry := true
                 else
                  if clflag <> true  then
                    begin
                       entry:=false;
                       g2:=g2+(span-distance)*intensity;
                       g3 := g2 -g1 ;
                       for i:= trunc(round(x*100/length))+1
                                to 100 do
                           c[i,8] := g1;
                       for i:= trunc(round(y*100/length))+1
                                to 100 do
                           c[i,8] :=c[i,8]+g3;
                       sh1:=0;sh2:=0;
                       for i:= 0 to 100 do
                           begin
                                c[i,6] :=c[i,6]+c[i,8]-c[i,7];
                                if c[i,6]>sh2 then sh2:=c[i,6];
                                if c[i,6]<sh1 then sh1:=c[i,6];
                           end;
                    end;

            end;
      end;
    end;
    for i:=0 to 100 do c[i,9]:=c[i,9]+c[i,5];

-----------------------------------------------------------------}

l1:=0;l2:=0;
for i:= 0 to 99 do
                begin
                   if c[i,5]< l1 then l1:=c[i,5];
                   if c[i,5]> l2 then l2:=c[i,5];
                end;
assign(fil1,'data.dat');
rewrite(fil1);
writeln(fil1,bmtype);
writeln(fil1,length*1000.0);
writeln(fil1,concgrd);
writeln(fil1,grade);
writeln(fil1,l1*10000000.0);
writeln(fil1,l2*10000000.0);
writeln(fil1,sh1*1000000.0);
writeln(fil1,sh2*1000000.0);
writeln(fil1,stldia*10);
writeln(fil1,depth*1000.0);
writeln(fil1,breadth*1000.0);
close(fil1);
{-----------------------------------------------------------   }
```

```
clrscr;
write('Would you like to see the graphical output[y/n] :');
read(answere);
     if ((answere='y')OR(answere='Y'))then
        begin
          graphcolormode;
             r:=0;
             for i:= 0 to 100 do
                   if abs( c[i,5]) > r then r:=abs( c[i,5]);
             if r<>0 then
             begin
                j:=48;
                draw (50,50,248,50,1);
                for i:= 1 to 100  do
                   begin
                       l:= trunc(round(c[i,5]/ r * 40.0));
                       j:=j+2;
                       k:=50-l  ;
                       draw(j,50,j,k,2)
                   end;
            if fxflag=true then
              begin
                  j:=49;
                  draw (50,90,250,90,1);
                  for i:= 0 to 100  do
                    begin
                        l:= trunc(round(g[i]/ r * 40.0));
                        j:=j+2;
                        k:=50-l  ;
                        draw(j,50,j,k,9);
                     end;
                   j:=49;
                   for i:= 0 to 100 do
                      begin
                           begin
                              l:= trunc(round(f[i]/ r * 40.0));
                              j:=j+2;
                              k:=90-l  ;
                              draw(j,90,j,k,2);
                          end;
                     end;
              end;
             r:=0;
             for i:= 1 to 100 do
                   if abs(c[i,6]) > r then r:=abs( c[i,6]);
             if r<>0 then
             begin
                j:=48;
                draw (50,160,248,160,1);
                for i:= 1 to 100  do
                   begin
                       l:= trunc(round(c[i,6]/ r * 40.0));
                       j:=j+2;
                       k:=160-l  ;
                      draw(j,160,j,k,2)
```

```
              end;
        end;
      write('Like to see the combined graphical [y/n] :');
      read(answere);
      if ((answere='y')OR(answere='Y'))then
        begin
          clrscr;
          graphcolormode;
           r:=0;
            for i:= 0 to 100 do
              if abs( c[i,9]) > r then r:=abs( c[i,9]);
              if r<>0 then
                begin
                  j:=48;
                  draw (50,50,248,50,1);
                  for i:= 1 to 100  do
                     begin
                        l:= trunc(round(c[i,9]/ r * 40.0));
                        j:=j+2;
                        k:=50-l ;
                        draw(j,50,j,k,2)
                     end;
                end;
        end;
      end;
   end;
end.
```

PROGRAM DESIGN.PRO

```
code=2635
nowarnings
domains
        file=mf
database
        beam(string)
        steel(integer)
        stldia(real)
        mindepth(real)
        safe_depth(real)
        safe_breadth(real)
        Bmfac(real)
        length(real)
        pbmd(real)
        nbmd(real)
        design_moment(real)
        pshr(real)
        nshr(real)
        minbreadth(real)
        x_lim(real)
        mu_lim(real)
        m_load(real)
        pascald(real)
        pascalb(real)
        clear_depth(real)
        finalbreadth(real)
        finaldepth(real)
        eff_depth(real)
        cover(real)
        dpth_section(real)
        concrete(symbol)
        concgrd(real)

        stlgrd(real)
predicates
        start
        wr2(real,real,real)
        window
        option
        breadth
        dcd
        gm1(real,symbol)
        gm2(real,symbol)
        hm1(real,symbol)
        he2(real,sy-bo,)
        dcd1(symbol)
        dcd2(symbol)
        decide
        decide_b(real)
        decide_d(real)
        cvr(real)
        bmfactor
        roundit(real,real)
        gr(real,real,real)
```

```
        less(real,real,real)
        conc
        b(real)
        check(real,real,real)
        safe_section(real)
        stl
        stlrnf(real,real,real)
        design
        dsgn       (real,real,real,real,real,real,real)
        prelim_dsgn(real,real).
        max_prsnt_rnf_sngl(real,real,real)
        lim_mom_rest(real,real,real)
        modfac_tsn(real,real,real,real)
        modfac_comp(real,real)
        stlcal(real,real,real)
        decide_depth(real,real,real,real,real,real,real)
goal
    start .
clauses
        modfac_tsn(0.0,2.100,2.400,3.000).
        modfac_tsn(0.2,1.350,1.600,2.300).
        modfac_tsn(0.4,1.075,1.280,2.100).
        modfac_tsn(0.6,0.950,1.100,1.730).
        modfac_tsn(0.8,0.900,1.025,1.525).
        modfac_tsn(1.0,0.850,0.950,1.415).
        modfac_tsn(1.2,0.825,0.925,1.320).
        modfac_tsn(1.4,0.810,0.900,1.285).
        modfac_tsn(1.6,0.800,0.880,1.215).
        modfac_tsn(1.8,0.790,0.860,1.150).
        modfac_tsn(2.0,0.775,0.840,1.125).
        modfac_tsn(2.2,0.760,0.820,1.100).
        modfac_tsn(2.4,0.745,0.800,1.085).
        modfac_tsn(2.6,0.730,0.785,1.065).
        modfac_tsn(2.8,0.715,0.770,1.045).
        modfac_tsn(3.0,0.700,0.755,1.025).

        modfac_comp(0.00,0.00).
        modfac_comp(0.25,0.70).
        modfac_comp(0.50,1.14).

        modfac_comp(0.75,1.20).
        modfac_comp(1.00,1.24).
        modfac_comp(1.25,1.285).

        modfac_comp(1.50,1.33).
        modfac_comp(1.75,1.36).
        modfac_comp(2.00,1.40).

        modfac_comp(2.25,1.42).
        modfac_comp(2.50,1.455).
        modfac_comp(2.75,1.48).
        modfac_comp(3.00,1.51).

        max_prsnt_rnf_sngl(15,250,1.32).
        max_prsnt_rnf_sngl(15,415,0.72).
```

```
max_prsnt_rnf_sngl(15,500,0.57).

max_prsnt_rnf_sngl(20,250,1.76).
max_prsnt_rnf_sngl(20,415,0.96).
max_prsnt_rnf_sngl(20,500,0.76).

max_prsnt_rnf_sngl(25,250,2.20).
max_prsnt_rnf_sngl(25,415,1.19).
max_prsnt_rnf_sngl(25,500,0.94).

max_prsnt_rnf_sngl(30,250,2.64).
max_prsnt_rnf_sngl(30,415,1.43).
max_prsnt_rnf_sngl(30,500,1.13).

lim_mom_rest(15,250,2.24).
lim_mom_rest(15,415,2.07).
lim_mom_rest(15,500,2.00).

lim_mom_rest(20,250,2.98).
lim_mom_rest(20,415,2.76).
lim_mom_rest(20,500,2.66).

lim_mom_rest(25,250,3.73).
lim_mom_rest(25,415,3.45).
lim_mom_rest(25,500,3.33).

lim_mom_rest(30,250,4.47).
lim_mom_rest(30,415,4.14).
lim_mom_rest(30,500,3.99).

start :-    write("Lets start"),
            openread(mf,"data.dat"),readdevice(mf),
            readln(Beam),
            readreal(LENGTH),
            readln(CC),
            readint(SS),
            readreal(NB),Nbmd=1.5*Nb,
            readreal(PB),Pbmd=Pb*1.5,
            readreal(Sh1),S1=Sh1*1.5,
            readreal(Sh2),S2=Sh2*1.5,
            readreal(Stldia),
            readreal(Depth),
            readreal(Breadth),
            closefile(mf),
            asserta(beam(Beam)),
            asserta(concrete(CC)),
            asserta(steel(SS)),
            asserta(nshr(Sh1)),
            asserta(pshr(Sh2)),
            asserta(length(LENGTH)),
            asserta(nbmd(NBMD)),
            asserta(pbmd(PBMD)),
            asserta(pascald(Depth)),
            asserta(pascalb(Breadth)),
            asserta(stldia(Stldia)),window,
```

```
                bmfactor,!,conc,!,stl,!,cvr(Stldia),
                safe_section(Length),option,breadth,
                design.

window   :- makewindow(2,1,112,"Auto Beam",0,0,25,80),nl,nl,nl.
option   :- write("     This expert system by default designs"),
            write(" a singly reinforced beam, if you  do not "),
            write("specify dimensions of the beam section."),
            write("but it opts for a"),nl,
            write("doubly reinforced beam if the section you"),
            write(" specified cannot safely handle the loads.").
conc     :- concrete(m10),asserta(concgrd(10)).
conc     :- concrete(m15),asserta(concgrd(15)).
conc     :- concrete(m20),asserta(concgrd(20)).
conc     :- concrete(m25),asserta(concgrd(25)).
conc     :- concrete(m30),asserta(concgrd(30)).
conc     :- concrete(m35),asserta(concgrd(35)).
conc     :- concrete(m40),asserta(concgrd(40)).
conc     :- concrete(m50),asserta(concgrd(45)).

stl      :- steel(1),asserta(stlgrd(500)),A=(805/(1265+500)),
            asserta(x_lim(A)).
stl      :- steel(2),asserta(stlgrd(415)),A=(805/(1265+415)),
            asserta(x_lim(A)).
stl      :- steel(3),asserta(stlgrd(250)),A=(805/(1265+250)),
            asserta(x_lim(A)).

bmfactor :- beam(cl),asserta(bmfac( 7.0)).
bmfactor :- beam(ss),asserta(bmfac(20.0)).
bmfactor :- beam(fx),asserta(bmfac(26.0)).
bmfactor :- beam(oh),asserta(bmfac(26.0)).

safe_section(L):-bmfac(F),D=L/F,B1=L/60,B2=sqrt(L*D/250),
      gr(B1,B2,B3),write("     safe D,B " ,D,"   ",B3),nl,
      asserta(safe_depth(D)),asserta(safe_breadth(B3)).

cvr(Dia)  :- Dia)25,asserta(cover(Dia)).
cvr(Dia)  :- asserta(cover(25)).

breadth:-
concgrd(FCK),pascald(D),x_lim(X),stldia(SD),cover(C),pbmd(M1),
nbmd(M2),M3=abs(M2),gr(M1,M3,M),
asserta(design_moment(M)),nl,
write("      There can be infinite sections which can safely "),
write("handle the loads  here         are few and opt for one"),nl,nl,
write("      The moment induced by the specified loading :",M),nl,nl,
write(" Eff_D/Breadth   Clear Depth    Breadth  Eff_depth    Lim_Moment "),
!, decide_depth(FCK,D,X,SD,C,M,1.5),!.
decide_depth(FCK,D,X,SD,C,M,1.5):- !,
nl,Bd2= (1.5*M) / (0.36*FCK*(1-(0.42*X))*X),   D1=ln(Bd2)/3,
D2=exp(D1), D4=D2+C+(Sd/2),Dm=D4/10,
roundit(Dm,Dn),D5=10*Dn,D6=round(D5-C-(Sd/2)),
D7=round(D5/1.5),
Mom = (0.36*Fck*D7*D6*D6*( 1-(0.42*X))*X),
write("      1.5 ","              ",D5,"               ",D7,"      ",D6,"            ",Mom),nl,
```

```
decide_depth(FCK,D,X,SD,C,M,2),!.
decide_depth(FCK,D,X,SD,C,M,2):-  !,
Bd2= 2*M / (0.36*FCK*(1-(0.42*X))*X),   D1=ln(Bd2)/3,
D2=exp(D1), D4=D2+C+(Sd/2),Dm=D4/100,
Roundit(Dm,Dn),D5=100*Dn,D6=round(D5-C-(Sd/2)),
D7=round(D6/2),Mom =( 0.36*Fck*D6*D6*D6*( 1-(0.42*X))*X/2),
write("       2.0 ","             ",D5,"               ",D7,"      ",D6,"              ",Mom),nl,
decide_depth(FCK,D,X,SD,C,M,2.5),!.
decide_depth(FCK,D,X,SD,C,M,2.5):-  !,
Bd2=2.5*M / (0.36*FCK*(1-(0.42*X))*X),   D1=ln(Bd2)/3,
D2=exp(D1), D4=D2+C+(Sd/2),Dm=D4/100,
roundit(Dm,Dn),D5=100*Dn,D6=round(D5-C-(Sd/2)),
D7=round(D6/2.5),Mom =( 0.36*Fck*D6*D6*D6*( 1-(0.42*X))*X/2.5),
write("       2.5 ","             ",D5,"               ",D7,"      ",D6,"              ",Mom),nl,
decide_depth(FCK,D,X,SD,C,M,3),!.
decide_depth(FCK,D,X,SD,C,M,3):-  !,
Bd2= 3*M / (0.36*FCK*(1-(0.42*X))*X),   D1=ln(Bd2)/3,
D2=exp(D1), D4=D2+C+(Sd/2),Dm=D4/100,
roundit(Dm,Dn),D5=100*Dn,D6=round(D5-C-(Sd/2)),
D7=round(D6/3),Mom =(0.36*Fck*D6*D6*D6*( 1-(0.42*X))*X/3),
write("       3.0 ","             ",D5,"               ",D7,"      ",D6,"              ",Mom),nl,
decide_depth(FCK,D,X,SD,C,M,3.5),!.
decide_depth(FCK,D,X,SD,C,M,3.5):-  !,
Bd2= 3.5*M / (0.36*FCK*(1-(0.42*X))*X),   D1=ln(Bd2)/3,
D2=exp(D1), D4=D2+C+(Sd/2),Dm=D4/100,
roundit(Dm,Dn),D5=100*Dn,D6=round(D5-C-(Sd/2)),
D7=round(D6/3.5),Mom =(0.36*Fck*D6*D6*D6*( 1-(0.42*X))*X/3.5),
write("       3.5 ","             ",D5,"               ",D7,"      ",D6,"              ",Mom),nl,
decide_depth(Fck,D,X,SD,C,M,4),!.
decide_depth(FCK,D,X,SD,C,M,4):-  !,
Bd2= 4*M / (0.36*FCK*(1-(0.42*X))*X),   D1=ln(Bd2)/3,
D2=exp(D1), D4=D2+C+(Sd/2),Dm=D4/100,
roundit(Dm,Dn),D5=100*Dn,D6=round(D5-C-(Sd/2)),
D7=round(D6/4),Mom =(0.36*Fck*D6*D6*D6*( 1-(0.42*X))*X/4),
write("       4.0 ","             ",D5,"               ",D7,"      ",D6,"              ",Mom),nl,
dcd,!.

dcd     :-
write("     Do you want to specify breadth or depth or"),
write(" both dimensions of the beam        section [y/n]:"),
readln(Reply),dcd1(Reply),!.
dcd1(R) :-
R="n",nl,nl,write("     This systems design is based on "),
write("the IS-456 codal specifications"),safe_depth(D),
safe_breadth(B),prelim_dsgn(B,D).
dcd1(R) :- R="y",
  write("     Enter    'b'  to specify  Breadth"),nl,
  write("     Enter    'd'  to specify  Depth   "),nl,
  write("     Enter    'bo' to specify  Both    "),nl,
  write("     d / b / bo : "),readln(Reply),dcd2(Reply).
dcd2(R) :-
  R="d",write("     Enter the depth[Mm]: "),readreal(D1),nl,
  decide_b(D1).
dcd2(R) :- R="b",write("     Enter the breadth[Mm]: "),
   readreal(B), decide_d(B).
```

```
dcd2(R)  :- R="bo",decide.

decide_d(B):-
        length(L),design_moment(M),concgrd(FCK),x_lim(X),
        stldia(SD),cover(C),safe_depth(Safe),
         D = sqrt(M /  (0.36*FCK*B*(1-(0.42*X))*X)),
         gr(D,Safe,D1),D2=D1/10,  roundit(D2,D3),
         D4=D3*10,D5=D4-C-(Sd/2),
         Mom =(0.36*Fck*D5*D5*B*( 1-(0.42*X))*X),
         write("      The moment induced by loads : ",M," N-mm"),nl,
         write("      Breadth you specified[Mm] :",B),nl,
         write("      Depth which can sustain the specifed"),
         write(" loading[Mm] :",D),nl,
         write("      The depth satisfying IS-456  ",Safe),nl,
         write("      The greater eff-depth  ",D1,"  cosidered ."),nl,
         write("      The moment of resistence :",Mom),nl,Mm=Mom/M,
         write("      The factor of safety  ",Mm),nl,
         write("      Is this section satisfactory to you [y/n]: "),
         readln(Reply),
         hm1(B,Reply),   prelim_dsgn(B,D4).

 decide_b(D)  :- safe_depth(Safe),S1=Safe/4,D<=S1,
        write("      The depth given is not at all safe"),
        write("for a singly reinfofrced beam.").

 decide_b(D)  :-length(L),design_moment(M),concgrd(FCK),x_lim(X),
         stldia(SD),cover(C),D1=D-C-(Sd/2),safe_depth(Safe),
         B = M /  (0.36*FCK*D1*D1*(1-(0.42*X))*X),Bb=B/10,
         roundit(Bb,B1),B2=B1*10,
         Bm=L/60,Bn=sqrt(L*D/250),gr(Bm,Bn,B3),gr(B2,B3,B4),
         Mom =(0.36*Fck*D1*D1*B4*( 1-(0.42*X))*X),
         write("      The moment induced by loads : ",M," N-mm"),nl,
         write("      Depth you specified[Mm] :",D),nl,
         write("      Breadth which can sustain the specifed"),
         write(" loading[Mm] :",B),nl,
         write("      The breadth satisfying IS-456  ",B4),nl,
         write("      The greater breadth is  cosidered ."),nl,
         write("      The moment of resistence :",Mom),nl,Mm=Mom/M,
         write("      The factor of safety  ",Mm),nl,
         write("      Is this section satisfactory to you[y/n]: "),
         readln(Reply),gm1(D1,Reply).

 hm1(B,R)  :- R="y",
   write("      Thank you for concidering my advice."),nl.
 hm1(B,R)  :- R="n",
  write("      Which has to be changed breadth or depth [b/d]: "),
         readln(A),hm2(B,A).
 hm2(B,A)  :- A="d",write("      Enter the depth you prefer[mm] :"),
         readreal(D),prelim_dsgn(B,D).
 hm2(B,A)  :- A="b",dcd2("b").

 gm1(D,R)  :-
   R="y",write("      Thanks for considering my advice. " ),nl.
 gm1(D,R)  :- R="n",
  write("      Which has to be changed breadth or depth [b/d]: "),
```

```
        readln(A),gm2(D,A).
 gm2(D,A)  :-
        A="b",write("     Enter the breadth you prefer[mm] :"),
        readreal(B),prelim_dsgn(B,D).
 gm2(D,R)  :- R="d",dcd2("d").

wr2(B1,B,B2)  :- safe_breadth(B),gr(B,B1,B2).

decide  :-
  pascald(D),write("     The expert is of the opinion that "),
  write(" the depth should in no case should be     less "),
  write("than ",D," this depth doesnot include "),
  write("modification           factor  the minimum depth "),
  write(" can be further reduced if the modification        "),
  write("     factor is considered "),nl,nl,
  write("     The expert also wants to advice you that narrow"),
  write(" and deep beam are             economical and posses"),
  write(" greater stiffness and better lateral stability "),nl,
  write("     Now please enter your preferable depth and "),
  write("breadth use the above data         guidelines"),nl,
  write("     Enter the  clear depth you prefer : "),
  readreal(Reply),b(Reply).

    b(D)    :- length(L),B1=L/60,B2=sqrt(L*D/250),gr(B1,B2,B3),
    write("     The breadth should not be less than ",B3),nl,
    write("     Please enter the breadth you prefer :"),
    readreal(B5),check(B3,B5,D).
    check(Mb,B,D)  :-
       B<Mb,write("     This depth violates IS-456 codal"),
       write(" specifications, Try again "),
       nl,b(D).
check(M,B,D)  :- B1=D/B,  B1<2.0,
  write("     The breadth depth ratio is "),
  write("less than 2.0 ,The section may not be economical "),
  nl,b(D).
check(Mb,B,D)  :- B1=D/B,B1>5,
  write("     The depth breadth ratio is too"),
            write(" much please try again"),nl,b(D).
check(Mb,B,D)  :-
  write("     All the codal specifications regarding "),
            write("the depth and breadth are satisfied"),
            !,dsgn(M,Fck,X,Sd,C,B,D).
prelim_dsgn(B,D):-write(B,D).
dsgn(B,D)    :-concgrd(F),x_lim(X),design_moment(M),
    write("     Effective depth                     : ",D),nl,
    write("     Effective breadth                   : ",B),nl,
    Mlim= 0.36*F*B*D*D*( 1-(0.42*X))*X,nl,
    write("     Limimting moment of the section : ",Mlim),nl,
    Mlim>M,
   stlcal(D,M,F).
 dsgn(M,Fck,X,Sd,C,B,D):-
   safe_depth(D),safe_breadth(B),D1=D/10,B1=B/10,
   roundit(D1,D2),roundit(B1,B2),B3=B2*10,D3=D2*10,!,
   dsgn(M,Fck,X,Sd,C,B3,D3).
   design:-write("    The design starts.").
```

```
stlcal(D,M,Fck):- stlgrd(Fy),A1=-Fck*D*D/2,
   A2=M*D*Fck/2*(0.87*Fy),R1=(-A1+sqrt(A1*A1-4*Fy*A2))/(2*Fy),
   R2=(-A1-sqrt(A1*A1-4*Fy*A2))/(2*Fy),
   write("      Roots of the equation        ",R1,"    ",R2).
                  stlrnf(A,B,D).

 stlrnf(A,B,D):-
  readln(Reply),Reply="y";Reply="Y",
  write("      Enter the dia of bars"),
  readreal(Fai)./* decice the bars and number*/

gr(A,B,C) :- A>=B,C=A.
gr(A,B,C) :- C=B.

less(A,B,C)  :- A <= B, C=A.
less(A,B,C)  :- C=B.

roundit(A,B) :- C=round(A)-A, C>=0, B=round(A).
roundit(A,B) :- B=round(A)+0.5.
```

ACKNOWLEDGEMENT

I am very much indebted to my guide, **Dr. S. Balasundaram,** Asst. Professor, who has been extremely helpful and encouraging through out the project, without which it would have been very difficult to complete the project.

Dr. Ashok Gupta, Asst. Professor, Indian Institute of Technology, New Delhi, has played a very significant role by giving his timely and extremely useful suggestions and sparing his precious time disscussing with us.

I thank **Mr. D. Jagadesh, Mr. Seshu, Mr. Pinakapani** for their help and advices.

I also thank **Mr. Fateh Singh,** Administrative officer of our school for his cooperation.

I thank our dean **Prof. Karmeshu** , who is very encouraging through out this project.

Jagadesh. P. Nanisetty