ARTIFICIAL INTELLIGENCE (3170716)

VVP ENGINEERING COLLEGE

SUBMITTED BY: **DISHEN MAKWANA**

180470107035

G2

V. V. P. Engineering College, Rajkot

Department of Computer Engineering

Vision of the Institute

- To be an exemplary institute, transforming students into competent professionals with human values.

Mission of the Institute

- To provide a conducive academic environment for strengthening technical capabilities of the students.
- To strengthen linkage with industries, alumni and professional bodies.
- To organize various co-curricular and extra-curricular activities for overall development of the students.
- To practice good governance and conduct value- based activities for making students responsible citizens.

Vision of the Department

- Transforming students into globally efficient professionals with moral values.

Mission of the Department

- To provide a strong foundation of computer engineering through effective teaching learning process.
- To enhance industry linkage & alumni network for better placement and real-world exposure.

- To provide various opportunities & platforms for all round development of students & encourage them for value-based practices.

Program Educational Objectives (PEOs)

Graduates will be able to

- Apply computer engineering theories, principles and skills to meet the challenges of the society.
- Communicate effectively, work collaboratively and manifest professionalism with ethics.
- Exhibit life-long learning attitude and adapt to rapid technological changes in industry.
- Advance their career in industry, pursue higher education or become an entrepreneur.

राष्ट्राय स्वाहा इदं न मम।

V.V.P. ENGINEERING COLLEGE

RAJKOT

Certificate

This is to certify that

Mr. DISHEN MAKWANA, Enrollment No: 180470107035, Branch: Computer Engineering, Semester: 7 has satisfactorily completed the course in the subject: **ARTIFICIAL INTELLIGENCE (3170716)** within the four walls of V.V.P. Engineering College, Rajkot.

Date of Submission:

Prof. Komil Vora,	Head of Department,
Staff In-Charge	Department of Computer Engineering,
	V.V.P. Engineering College

V. V. P. Engineering College
Department of Computer Engineering
Course Outcomes

Semester: 7th

Subject: Artificial Intelligence

Code: 3170716

After learning the course, the students will be able to

CO Number	Course Outcomes	CL
C70716.1	Apply basic principles of AI in problem solving	Ap
C70716.2	Evaluate knowledge representation issues using AI rules.	E
C70716.3	Analyse working of reasoning in presence of incomplete and/or uncertain information.	A
C70716.4	Demonstrate awareness of AI techniques in intelligent agents, expert systems, artificial neural networks and other machine learning models.	U
C70716.5	Apply Prolog Programming language for Basic AI problems.	Ap

Index

EXP NO.	EXPERIMENT	NO. OF LAB	Sign
1	1.What is prolog?	LAB 1	
	2.Difference between OOL and procedural Languages.		
	3.Benefits of using prolog		
	4.Feature of prolog		
	5.Application of prolog.		
2	1.Write a program for the following task,	LAB 2 & LAB 3	
	1)Students who are living in Rajkot.		
	2)Students whose age is greater than 15 and living in Rajkot.		
	3)Student who has more than 60%.		
	2.Write a program for the family tree.		
3	1.Write a program to check whether given character is a character or digit.	LAB 4 & LAB 5	
	2.Write a program to generate a random number with respect to entered digit.		
	3.Write a program to implement Calculator recursively.		
4	1.Write a program to implement login system.	LAB 6	
	2.Write a program to implement login system recursively.		
	3.Write a program to implement login system using repeat predicate.		
5	1.Write a program to display the element of given list.	LAB 7	
	2.Write a program to check whether the given element is in the list or not.		
	3.Write a program to find the largest number from a given list.		
6	1.Write a program to print last element of the list.	LAB 8	
	2.Write a program to find length of list.		
	3.Write a program to generate sum of elements of given list.		
7	1.Write a program to union of two lists.	LAB 9 & LAB10	
	2.Write a program to reverse a given list.		
8	1.Write a program to convert integer list to equivalent symbol list.	LAB 11 & LAB 12	
	2.Write a program to create, read, write and delete a file		
9	Write a program to Calculate factorial of N	LAB13 & LAB14	
10	Write a Prolog program to check whether a given list is palindrome or not		
11	Write a Prolog Program for Monkey and banana problem	LAB15	

LAB 1

1. What is prolog?

- Prolog stands for programming in logic. In the logic programming paradigm, prolog language is most widely available. Prolog is a declarative language, which means that a program consists of data based on the facts and rules (Logical relationship) rather than computing how to find a solution. A logical relationship describes the relationships which hold for the given application.
- To obtain the solution, the user asks a question rather than running a program. When a user asks a question, then to determine the answer, the run time system searches through the database of facts and rules.
- The first Prolog was 'Marseille Prolog', which is based on work by Colmerauer. The major example of fourth-generation programming language was prolog. It supports the declarative programming paradigm.
- In 1981, a Japanese computer Project of 5th generation was announced. After that, it was adopted Prolog as a development language. In this tutorial, the program was written in the 'Standard' Edinburgh Prolog. Prologs of PrologII family are the other kind of prologs which are descendants of Marseille Prolog.
- Prolog features are 'Logical variable', which means that they behave like uniform data structure, a backtracking strategy to search for proofs, a pattern-matching facility, mathematical variable, and input and out are interchangeable.
- To deduce the answer, there will be more than one way. In such case, the run time system will be asked to find another solution. To generate another solution, use the backtracking strategy. Prolog is a weakly typed language with static scope rules and dynamic type checking.
- Prolog is a declarative language that means we can specify what problem we want to solve rather than how to solve it.

2. Difference between OOL and procedural Languages.

Procedural Oriented Programming	Object Oriented Programming
In procedural programming, program is divided into small parts called **functions**.	In object oriented programming, program is divided into small parts called **objects**.
Procedural programming follows **top down approach**.	Object oriented programming follows **bottom up approach**.
There is no access specifier in procedural programming.	Object oriented programming have access specifiers like private, public, protected etc.
Adding new data and function is not easy.	Adding new data and function is easy.
Procedural programming does not have any proper way for hiding data so it is **less secure**.	Object oriented programming provides data hiding so it is **more secure**.

Procedural Oriented Programming	Object Oriented Programming
In procedural programming, overloading is not possible.	Overloading is possible in object oriented programming.
In procedural programming, function is more important than data.	In object oriented programming, data is more important than function.
Procedural programming is based on **unreal world**.	Object oriented programming is based on **real world**.
Examples: C, FORTRAN, Pascal, Basic etc.	Examples: C++, Java, Python, C# etc.

3. Application of prolog.

- Prolog is used in some areas like database, natural language processing, artificial intelligence, but it is pretty useless in some areas like a numerical algorithm or instance graphics.
- In artificial intelligence applications, prolog is used. The artificial intelligence applications can be automated reasoning systems, natural language interfaces, and expert systems. The expert system consists of an interface engine and a database of facts. The prolog's run time system provides the service of an interface engine.
- A basic logic programming environment has no literal values. An identifier with upper case letters and other identifiers denote variables. Identifiers that start with lower-case letters denote data values. The basic Prolog elements are typeless. The most implementations of prolog have been enhanced to include integer value, characters, and operations. The Mechanism of prolog describes the tuples and lists.
- Functional programming language and prolog have some similarities like Hugs. A logic program is used to consist of relation definition. A functional programming language is used to consist of a sequence of function definitions. Both the logical programming and functional programming rely heavily on recursive definitions.
- Specification Language
- Robot Planning
- Natural language understanding
- Machine Learning
- Problem Solving
- Intelligent Database retrieval
- Expert System
- Automated Reasoning

4. Feature of prolog

The main characteristics/notions of the Visual Prolog programming language are:

- based on logical programming with Horn clauses
- fully object oriented
- object predicate values (delegates)
- strongly typed
- algebraic data types

- pattern matching and unification
- controlled non-determinism
- fully integrated fact databases
- supports parametric polymorphism
- automatic memory management
- supports direct linkage with C/C++
- supports direct calling of Win32 API functions

5. Benefits of using Prolog.

- Easy to build database. Doesn't need a lot of programming effort.
- Pattern matching is easy. Search is recursion based.
- It has built in list handling. Makes it easier to play with any algorithm involving lists.

LAB 2 & LAB 3

1. Write a program for the following task,

1) Students who are living in Rajkot.
2) Students whose age is greater then 15 and living in Rajkot.
3) Student who has more than 60%.

Answer:

```
domains
predicates
        student(symbol,symbol,integer,integer)
clauses
        student("Sita","Rajkot",20,70).
        student("Rita","Delhi",35,55).
        student("Nita","Bangalore",19,90).
        student("Mita","Rajkot",55,30).
        student("Hita","Mumbai",60,67).
goal
/*      student(Name,"Rajkot",Age,Percentage).*/
/*      student(Name,"Rajkot",Age,Percentage), Age>50.*/
/*      write("Students with percentage greater than 60").*/
        student(Name,City,Age,Percentage), Percentage > 60.
```

2. Write a program for the family tree.

```
                       Rohan-Rina
  ____________________________|_____________________
  |              |              |              |
Nita-Numa     Gita-Gautam    Sonu-Sonia     Monu-Mina                    |
                 |
____|____     ____|______                   ____|_____
|    |  |      |               |    |
Rosh Posh     Sam    Pam                    John Jenny
```

Answer:

domains

```
        X,Y = String
predicates
        male(String)
        female(String)
        parent(String,String)
        father(String,String)
        mother(String,String)
        brother(String,String)
        sister(String,String)
        grandfather(String,String)
        grandmother(String,String)
        uncle(String,String)
        uncle1(String,String)
        aunt(String,String)
        aunt1(String,String)
clauses
        male("Rohan").
        male("Sonu").
        male("Monu").
        male("Sam").
        male("John").
        male("Gautam").
        male("Numa").
        male("Rosh").
        male("Posh").
        female("Rina").
        female("Nita").
        female("Gita").
        female("Pam").
        female("Jenny").
        female("Mina").
        female("Sonia").
        parent("Nita","Rohan").
        parent("Gita","Rohan").
        parent("Sonu","Rohan").
        parent("Monu","Rohan").
        parent("Nita","Rina").
        parent("Gita","Rina").
        parent("Sonu","Rina").
        parent("Monu","Rina").
        parent("Rosh","Numa").
        parent("Posh","Numa").
        parent("Rosh","Nita").
        parent("Posh","Nita").
        parent("Sam","Gautam").
        parent("Pam","Gautam").
        parent("Sam","Gita").
        parent("Pam","Gita").
        parent("","Sonu").
        parent("","Sonia").
        parent("John","Monu").
        parent("Jenny","Monu").
        parent("John","Mina").
        parent("Jenny","Mina").

        mother(X,Y):-
                female(Y),
                parent(X,Y).

        brother(X,Y):-
                male(Y),
                father(X,Z),
                father(Y,Z),X<>Y.

        sister(X,Y):-
                female(Y),
                father(X,Z),
                father(Y,Z),X<>Y.

        grandfather(X,Y):-
```

```
                male(Y),
                parent(X,Z),
                parent(Z,Y).

        grandmother(X,Y):-
                female(Y),
                parent(X,Z),
                parent(Z,Y).

        uncle1(X,Y):-
                male(Y),
                parent(X,Z),
                brother(Z,Y).

        uncle(X,Y):-
                uncle1(X,Y);
                aunt1(X,Z),
                mother(P,Z),
                father(P,Y).

        aunt1(X,Y):-
                female(Y),
                parent(X,Z),
                sister(Z,Y).

        aunt(X,Y):-
                aunt1(X,Y);
                uncle1(X,Z),
                parent(P,Z),
                mother(P,Y).
        father(X,Y):-
                male(Y),
                parent(X,Y).
goal
                aunt1(X,Y).
```

LAB 4 & LAB 5

1. Write a program to check whether given character is a character or digit.

```
domains
predicates
        getinput()
        check(integer)
clauses
        getinput():-
                readchar(X),
                write("Input is ",X),
                char_int(X,Y),
                check(Y).
        check(Y):-
                Y>=97,
                Y<=122,
                write("\nIt is a lowercase character\n").
        check(Y):-
                Y>=65,
                Y<=91,
                write("\nIt is a uppercase character\n").

        check(Y):-
                Y>=47,
                Y<=56,
                write("\nIt is a digit\n").
goal
        getinput();
        write("\nIt is neither a digit nor a character\n").
```

2. Write a program to generate a random number with respect to entered digit.

```
domains
predicates
```

```
        getinput()
clauses
        getinput():-
                readint(X),
                write("Input is ",X),
                X>0,
                random(X,Y),
                write("\nRandom number is ",Y,"\n").
goal
        getinput();
        write("\nInput is invalid\n").
```

3. Write a program to implement Calculator recursively.

```
domains
predicates
        getinput()
        check(integer)
        calculator(integer,integer,integer)
clauses
        getinput():-
                write("\n1)ADDITION \n2)SUBTRACTION \n3)MULTIPLICATIN \n4)DIVISION
\n5)EXIT \n\nEnter your choice:-"),
                readint(C),
                check(C),
                write("\nEnter first number: "),
                readint(X),
                write("Enter second number: "),
                readint(Y),
                calculator(C,X,Y),
                getinput(),
                write("\n");
                write("\nInput is invalid please try again\n"),
                getinput().

        check(C):-
```

```
                C=5,
                exit(1);
                C>5,
                write("\nInput is invalid please try again\n"),
                getinput();
                C<5.

        calculator(1,X,Y):-
                Z=X+Y,
                write("\nAddition is ",Z,"\n").
        calculator(2,X,Y):-
                Z=X-Y,
                write("\nSubtraction is ",Z,"\n").
        calculator(3,X,Y):-
                Z=X*Y,
                write("\nMultiplication is ",Z,"\n").
        calculator(4,X,Y):-
                Y<>0,
                Z=X/Y,
                write("\nDivision is ",Z,"\n");
                write("\nDivisin by zero is not allowed\n").
/*      calculator(C,X,Y):-
                write("\nInvalid choice\n").
        */
goal
        getinput().
```

LAB 6

1.Write a program to implement login system.

```
domains
        Name, Password = symbol
predicates
        getinput(Name,Password).
        user(Name, Password).
        login.
clauses
        getinput(Name,Password):-
                write(“Enter User Name : ”),nl,
                readln(Name),nl,
                write(“Enter Password”),nl,
                readln(Password),nl,
                user(Name,Password).
                user(“abc”,”123”).
                user(“def”,”456”).

                login:-
                        getinput(Name,Password),nl,
                        write(“Login Successful.”).
                login:-
                        write(“Your are not authorized to login.”).
goal
        login.
```

2.Write a program to implement login system recursively.

```
domains
        Name, Password = symbol
predicates
        getinput(Name,Password).
        user(Name, Password).
        login.
```

```
clauses
        getinput(Name,Password):-
                write(“Enter User Name : ”),nl,
                readln(Name),nl,
                write(“Enter Password”),nl,
                readln(Password),nl,
                user(Name,Password).

                user(“abc”,”123”).
                user(“def”,”456”).
                login:-
                        getinput(Name,Password),nl,
                        write(“Login Successful.”).
                login:-
                        write(“Your are not authorized to login.”).
                        login.
goal
        login.
```

3.Write a program to implement login system using repeat predicate.

```
domains
        Name, Password = symbol
predicates
        getinput(Name,Password).
        user(Name, Password).
        login.
        Repeat.
clauses
                repeat.
                repeat:-
                repeat.
                getinput(Name,Password):-
                write(“Enter User Name : ”),nl,
                readln(Name),nl,
		write(“Enter Password”),nl,
		readln(Password),nl,
		user(Name,Password).

		user(“abc”,”123”).
		user(“def”,”456”).
		login:-
			repeat,
			getinput(_,_),nl,
			write(“Login Successful.”).
		login:-
			repeat,
			write(“Your are not authorized to login.  Try Again…”),
			getinput(_,_),nl,
			write(“Login Successful”).
goal
	login.
```

LAB 7

1.Write a program to display the element of given list.

```
domains
        list = symbol*
predicates
        go
        disp(list)
clauses
        go:-
                X = [abc,asdf,qwer],
                disp(X).

        disp([]).

        disp([H|T]):-
                write(H),
                nl,
                disp(T).
goal
        go.
```

2.Write a program to check whether the given element is in the list or not.

```
domains
        list = integer*

predicates
        go
        present(integer,list)
clauses
        go:-
                X = [1,2,3,0],
                write("Enter element: "),
                readint(I),
                write("Element = ",I),
                nl,
                present(I,X).

        present(I,[]):-
                T=[],
                write("Not present"),
                nl.

        present(I,[H|T]):-
                I=H,
                write("Present"),
                nl,
                exit(0);

                present(I,T).
goal
        go.
```

3.Write a program to find the largest number from a given list.

```
domains
        list = integer*

predicates
        go
        maximum(integer,list)
clauses
        go:-
                X = [1,7,3,6],
                maximum(-1,X).

        maximum(Max,[]):-
                T=[],
                write("Maximum number : ", Max),
                nl.

        maximum(Max,[H|T]):-
                Max < H,
                maximum(H,T);

                maximum(Max,T).
goal
        go.
```

LAB 8

1.Write a program to print last element of the list.

```
domains
        list = symbol*
predicates
        go
        last(list)
clauses
        go:-
                X = [abc,asdf,qwer],
                last(X).

        last([H|T]):-
                T=[],
                write("Last element is ",H),
                nl.

        last([H|T]):-
                last(T).
goal
        go.
```

2.Write a program to find length of list.

```
domains
        list = symbol*

predicates
        go
        count(integer,list)
clauses
        go:-
                X = [abc,asdf,qwer],
                count(0,X).
        count(C,[]):-
                T=[],
                write("Total number of elements : ",C),
                nl.
        count(C,[H|T]):-
                B = C + 1,
                count(B,T).
goal
        go.
```

3.Write a program to generate sum of elements of given list.

```
domains
        list = integer*
predicates
        go
        count(integer,list)
clauses
        go:-
                X = [1,2,3],
                count(0,X).

        count(Sum,[]):-
                T=[],
                write("Sum of elements : ",Sum),
                nl.
        count(Sum1,[H|T]):-
                Sum = Sum1 + H,
                count(Sum,T).
goal
        go.
```

LAB 9 & LAB 10

1.Write a program to union of two lists.

```
domains
        list = symbol*
predicates
        go
        union(list,list,list)
        disp(list)
clauses
        go:-
                X = [a1,b1,c1],
                Y = [d1,e1],
                Z = [],
                union(Z,X,Y).

        union(Z,[],[]):-
                disp(Z).
        union(Z,S,[H|T]):-
                N = [H|Z],
                union(N,S,T).
        union(Z,[H|T],S):-
                M = [H|Z],
                union(M,T,S).
        disp([]).
        disp([H|T]):-
                write(H),
                nl,
                disp(T).
goal
        go.
```

2.Write a program to reverse a given list.

```
domains
```

```
        list = integer*
predicates
        go
        rev(list,list)
        disp(list)
clauses
        go:-
                X = [1,2,3],
                Y = [],
                rev(X,Y).

        rev([],Y):-
                disp(Y).

        rev([H|T],Y):-
                M = [H|Y],
                nl,
                rev(T,M).
        disp([]).
        disp([H1|T1]):-
                write(H1),
                nl,
                disp(T1).
goal
        go.
```

LAB 11 & LAB 12

1.Write a program to convert integer list to equivalent symbol list.

```
domains
        ilist=integer*
        slist=symbol*
predicates
        go
        conv(ilist,slist)
        int_sym(integer,symbol)
clauses
        int_sym(0,zero).
        int_sym(1,one).
        int_sym(2,two).
        int_sym(3,three).
        int_sym(4,four).
        int_sym(5,five).
        int_sym(6,six).
        int_sym(7,seven).
        int_sym(8,eight).
        int_sym(9,nine).

        go:-
                I=[1,2,3,4,5,6,7],
                S=[],
                conv(I,S).
        conv([],S):-
                write(S).
        conv([H|T],S):-
                int_sym(H,A),
                M=[A|S],
                conv(T,M).
goal
        go.
```

2.Write a program to create, read, write and delete a file.

```
domains
        file=textFile
predicates
        go
        operation(integer)
        readfile
        writefile
clauses
        go:-
        disk("C:\\"),
        write("1.create"),nl,
        write("2.read"),nl,
        write("3.write"),nl,
        write("4.delete"),nl,
        write("5.exit"),nl,
        write("enter choice:"),
        readint(C),
        operation(C),
        go.

        operation(1):-
        write("enter Filename to create:"),
        readln(Name),
        openwrite(textFile,Name),
        writedevice(textFile),nl,
        closefile(textFile),
        writedevice(screen),
        write("File Sucessfully created.").

        operation(2):-
        write("enter Filename to read:"),
        readln(Name),
        openread(textFile,Name),
        readfile,
        closefile(textFile).

        operation(3):-
        write("enter Filename to Write:"),
        readln(Name),
        openwrite(textFile,Name),
        writefile,
        writedevice(screen),
        write("file successfully written.").

        operation(4):-
        write("Enter filename to delete:"),
        readln(Name),
        deletefile(Name),
        write("file successfully deleted"),nl.

        operation(_):-
        write("invalid choice."),nl.

        readfile:-
        readdevice(textFile),
        readln(L),
        writedevice(screen),
        readdevice(keyboard),
        write(L),nl,
        eof(textFile);
        readfile.

        writefile:-
        writedevice(screen),
        write("enter file content:"),
        readln(N),
        writedevice(textFile),
        write(N),nl,
          writedevice(screen),
          write("do you want to enter more?[y/n]"),
          readchar(C),
          C='y',
          writefile;
          C='n',
          closefile(textFile).
goal
          go.
```

LAB 13 & LAB 14

1.Write a program to Calculate factorial of N

```
predicates
   go
   find_factorial(integer,integer)
clauses
   go:-
      write("Enter a number = "),
      readint(Num),
      Result = 1,
      find_factorial(Num,Result).

   find_factorial(Num,Result):-
         Num <> 0,
      NewResult = Num * Result,
      NewNum = Num - 1,
      find_factorial(NewNum,NewResult).

   find_factorial(_,Result):-
      write("Factorial = ",Result),nl.

goal
   go.
```

.Write a Prolog program to check whether a given list is palindrome or not

```
domains
         list = integer*

predicates
         go
         rev(list,list,list)
         compare(list,list)
```

```
clauses
        go:-
                X = [1,2,3,3,2,2,1],
                Y = [],
                rev(X,Y,X).

        rev([],Y,X):-
                compare(Y,X).

        rev([H|T],Y,X):-
                M = [H|Y],
                nl,
                rev(T,M,X).

        compare([],[]):-
        write("\nList is Palindrome"),nl.

        compare([H|Y],[H|X]):-
        compare(Y,X).

        compare([H|Y],[K|X]):-
        write("\nList is not Palindrome"),nl.

goal
        go.
```

LAB 15

1.Write a Prolog Program for Monkey and banana problem.

```
domains
   state=state(symbol,symbol,symbol,symbol)
   /*state=state(monkey horizontal
            monkey vertical,
           box location,
           has/has not banana)    */
predicates
   move(state,symbol,state)
   canget(state)
clauses
   move(state(middle,onbox,middle,hasnot),
   grasp,state(middle,onbox,middle,has)).

   move(state(P,onfloor,P,hasnot),climb,
   state(P,onbox,P,hasnot)).

   move(state(P,onfloor,P,hasnot),push,
   state(P1,onfloor,P1,hasnot)).

   move(state(P1,onfloor,B,hasnot),walk,
   state(P2,onfloor,B,hasnot)).

   canget(state(_,_,_,has)) :-
      write(\"get\").
   canget(State1) :-
      move(State1,Move,State2),
      canget(State2),
      write(State2),nl.
goal
   clearwindow,
   canget(state(door,onfloor,window,hasnot))
```