PRACTICAL

FILE

Department: Computer Science and Engineering

Session: January - June

Subject: SLLP

Subject Code: CS-416

Semester: 8th

SRI SUKHMANI INSTITUTE OF ENGINEERING & TECHNOLOGY

Affiliated to PTU, & Approved by AICTE

Syllabus

1. Study of Propositional Logic

2. Study of First Order Predicate Logic

3. Introduction to prolog programming by a simple prolog program

4. Program to check whether input is alphabet or not

5. Program to find if given number is positive or negative.

6. Write a program to check whether a given person is a member of Club

7. Program illustrating the use of recursion that is finding sum of first N Integers.

8. Program for Bubble Sort

9. PROGRAM TO DETERMINE WHETHER A ELEMENT IS A MEMBER OF LIST in Artificial Intelligence

SRI SUKHMANI INSTITUTE OF ENGINEERING & TECHNOLOGY

Affiliated to PTU, & Approved by AICTE

List of Practical

Sr. No.	Topic
1.	Study of Propositional Logic
2.	Study of First Order Predicate Logic
3.	Introduction to prolog programming by a simple prolog program
4.	Program to check whether input is alphabet or not
5.	Program to find if given number is positive or negative.
6.	Write a program to check whether a given person is a member of Club
7.	Program illustrating the use of recursion that is finding sum of first N Integers.
8.	Program for Bubble Sort
9.	Program to determine whether a element is a member of list in Artificial Intelligence
10.	*Program to sort a list using Quick Sort in Artificial Intelligence

***Learning Beyond Syllabus Program to sort a list using Quick Sort in Artificial Intelligence**

Experiment 1

AIM: Study of Propositional Logic

Prolog:–

Prolog is a language which is used for programming in logic .prolog is computer programming language that is used for solving problem that involves object and the relationship between object .it is non procedural logic based language and not an algorithmic language .it is designed for natural language processing and has become one of the most widely used language for artificial intelligence. It involves building up knowledge base where relationship is represented.

Computer Programming consists of:-

a) Declaring some facts about object and their relationship.

b) Defining some rules about object and their relationship.

c) Asking question about object and their relationship**.**

We can consider prolog as a store house of facts and rules and its uses the fact and rules to another questions. Programming in prolog consists of supplying all these facts and rules.

It is conversational language which means that you and computer carry out a kind of conversation. The names of the object that are enclosed within the round brackets in each fact are called the arguments.

The name of the relationship which come first before the round brackets are called the predicate fact in prolog. simply allow you to empress arbitrary relationship between object .in prolog a question look just like a fact incepts that we put a general statement about object and their relationship . in prolog a rule consist of a head and a body is made up of a colon and hyphen .

The ":-"is pronounced as if.

1) Facts :- Rules for defining facts :

- The name of the all relationship and object must begin with a lower case letter. Eg likes (john.mary).
- The relationship is written first and then the object are written, separated by commas and the object are enclosed by the pair of round brackets.
- The full stop character must come at the end of the fact .

2) Question :-

Syntax: - ?- owns (Mary, book).

The question give ensure only in yes/no form.

3) Variables :- if u want to find out what things that john (person) likes, it is tiresome to ask , does john like books or does john like Mary and so on .

Instead of this we could ask the question of this as

Does john like x?

Syntax: - ?- likes (john,x).

Where x is variable and written in capital letter (uppercase letter)

Experiment 2

AIM:- Study of First Order Predicate Logic

FIRST ORDER LOGIC

- Where as propositional logic assumes the world contain facts
- First order logic (like natural language) assumes the world contain

1) **Object:** people, house. Number, colors, house, baseball, games, wars.
2) **Relation:** red, round, prime, brother, of bigger than part of ……
3) **Function:** father of, best friend, one more than plus…….

Syntax of foal: Basic elements

1) Constant: king john, 2, nus ….
2) Predicates: Brother, >…
3) Functions: sqrt, leftleg of ,……..
4) Variables: n,y,a,b,….
5) Connectives : >,<,<->
6) Equality :=
7) Quantifiers :<.-=

- **Truth in first order logic**

- Sentence are true with respect to a model and an interpretation .
- Model contains objects (domain elements) and relations among them.
- Interpretations specifies reference for

  Constant symbol – object

Predicate symbol – relations.

Function symbol –functional relation.

- **An atomic sentence predicate (term 1,….term n) if the object referred to buy term 1 ….term n are in the relation referred to buy predicate .**

- **Using fol**

- **The kinship domain :-**

- **Brother are siblings n,y brother (x,y)> sibling (n,y)**

- **Ones mother is ones female parent m,c mother © = m<>female (M)parent (M<C)**

- **Sibling is symmetric n,y sibling (n,y) –sibling (y.n)**

Experiment 3

AIM: Introduction to prolog programming by a simple prolog program

- You are given the following facts :

  Capital (“patna”,”bihar”).

  Capital (“lucknow”,”uttarpradesh”).

  Write goal for:

  1) the capital of the known state

  2) the state with its known capital

  3) list of state and their capital

- **domains**

  x= symbol

  y= symbol

- **Predicates**

  Capital (x,y)

- Clauses

  Capital (“patna”,”bihar”)

  Capital (“lucknow”,”uttarpradesh”)

OUTPUT:

Goal

Capital (x,"india")

X = delhi 1 solution

Capital ("bihar",y)

Y=bihar.

Experiment 4

AIM: Program to check whether input is alphabet or not

```
domains
list=integer*
predicates
findnum(integer,list)
clauses
findnum(X,[]):-
write("\nNumber Is Not Found").
findnum(X,[X|Tail]):-
write("\nNumber Is Found").
findnum(X,[Y|Tail]):-
findnum(X,Tail).
```

OUT PUT

=======

goal: findnum(3,[1,2,3,4,5])

Number Is Found

Yes

Goal: findnum(6,[1,2,3,4,5])

Number Is Not Found

Yes

Goal: findnum(2,[1,2,2,1])

Number Is Found

Yes

Experiment 5

AIM: Program to find if given nuber is positive or negative

Basic rule to check the number

If X>=0 then
X is positive
Else
X is negative

```
Domains
I= integer
Predicates
Pos_neg(i)
Clauses
Pos_neg(X):-X>=0, write("positive number"),nl.
Pos_neg(_ ):-write("negative number"),nl.
Goal
Pos_neg(4)
```

Output:

Positive number

Experiment 6

AIM: Write a program to check whether a given person is a member of Club

```
trace
domains
namelist = symbol*
N = symbol
predicates
member(symbol,namelist)
delete(symbol,namelist,namelist)
clauses
member(X,List):-
delete(X,List,_).
delete(X,[X|Tail],Tail).

delete(X,[Y|Tail1],[Y|Tail2]):-
delete(X,Tail1,Tail2).
```

Output

```
Goal: member(x,[a,b,x,z,
y])
Yes


Goal: member(t,[a,b,x,z,
y])
No
```

Experiment 7

AIM: Program illustrating the use of recursion that is finding sum of first N Integers

```
domains

    list=integer*


predicates

    findsum(list)
    sum(list,integer)


clauses

    findsum(L):-
        sum(L,Sum),
        write("\nSum Of Given List : ",Sum).

    sum([],0).

    sum([X|Tail],Sum):-
        sum(Tail,Temp),
        Sum=Temp+X.


OUT PUT
=======

Goal: findsum([1,2,3,4,5])

Sum Of Given List : 15

Yes

-------------------------------------
```

```
Goal: findsum([])


Sum Of Given List : 0
Yes

-------------------------------------

Goal: findsum([1,2,3,4,5,6,7,8,9,10])

Sum Of Given List : 55

Yes
```

Experiment 8

AIM: Program for Bubble Sort

```
Domains
    list = integer*.

Predicates
    bubblesort(list,list).
    swap(list,list).
    printlist(list).

Clauses
    bubblesort(InputList,SortList) :-
        swap(InputList,List) , ! ,
        printlist(List),
        bubblesort(List,SortList).
    bubblesort(SortList,SortList).

    swap([X,Y|List],[Y,X|List]) :- X > Y.
    swap([Z|List],[Z|List1]) :-
swap(List,List1).

    printlist([]) :- nl.
    printlist([Head|List]) :-
        write(Head, " "),
        printlist(List).

Output :

Goal: bubblesort([2,3,1,4],L).
2 1 3 4
1 2 3 4
L=[1,2,3,4]
1 Solution

Goal: bubblesort([2,4,1,3,5,9,6],L).
2 1 4 3 5 9 6
1 2 4 3 5 9 6
1 2 3 4 5 9 6
```

```
1 2 3 4 5 6 9
L=[1,2,3,4,5,6,9]
1 Solution
```

Experiment 9

AIM: PROGRAM TO DETERMINE WHETHER A ELEMENT IS A MEMBER OF LIST in Artificial Intelligence

```
trace
domains
    namelist = symbol*
     N = symbol
predicates
    member(symbol,namelist)
    delete(symbol,namelist,namelist)
clauses
    member(X,List):-
        delete(X,List,_).
    delete(X,[X|Tail],Tail).

    delete(X,[Y|Tail1],[Y|Tail2]):-
        delete(X,Tail1,Tail2).


Output

    Goal: member(x,[a,b,x,z,
    y])
    Yes


Goal: member(t,[a,b,x,z,
y])
No
```

Experiment 10

AIM: Program to sort a list using Quick Sort in Artificial Intelligence

```
Domains
list = integer*.

Predicates
quicksort(list,list).
split(integer,list,list,list).
concatenate(list,list,list).
printlist(list).

Clauses
quicksort([],[]).
quicksort([Head|Tail],SortedList) :-
split(Head,Tail,SList,BList),
quicksort(SList,SList1),
quicksort(BList,BList1),
concatenate(SList1,[Head|Blist1],SortedList),
printlist(SortedList).

split(_,[],[],[]).
split(Item,[Head1|Tail1],[Head1|SList],BList) :-
Item > Head1 , ! ,
split(Item,Tail1,SList,BList).
split(Item,[Head1|Tail1],SList,[Head1|BList]) :-
split(Item,Tail1,SList,BList).

concatenate([],List,List).
concatenate([Item|List1],List2,[Item|List3]) :-
concatenate(List1,List2,List3).

printlist([]) :- nl.
printlist([Head|Tail]) :-
write(Head," "),
printlist(Tail).
```

Output

```
Goal: quicksort([2,4,1,3,5,9,6],L).

1
3
6
6 9
5 6 9
3 4 5 6 9
1 2 3 4 5 6 9
L=[1,2,3,4,5,6,9]
1 Solution
```