ARTIFICIAL INTELLIGENCE LAB

(CS791)

Submitted by

ABHIJIT GHOSH

[CSE-4A Roll: 39]

(12015009001014)

Under supervision of:

Arunabha Tarafdar

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

UNIVERSITY OF ENGINEERING AND MANAGEMENT, KOLKATA

APRIL, 2018

INDEX

Sl.No.	Assignment	Signature	Remarks
1	Assignment 1		
2	Assignment 2		
3	Assignment 3		
4	Assignment 4		
5	Assignment 5		

ASSIGNMENT-1

1. *Code in prolog to compute the sum of a list.*

Code:

```
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.
```

Output:

```
      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
```

2. *Code in prolog to find maximum of 2 numbers.*

Code:

```
max(X,Y):-
(
 X=Y ->
  write('both are equal')
 ;
 X>Y ->
  (
```

```
  Z is X,
  write(Z)
  )
  ;
  (
  Z is Y,
  write(Z)
  )
).
```

Output:

```
Max(5,7).
7
```

3. *Code in prolog to find the length of the list.*

Code:

```
domains

   list=symbol*

predicates

   len(list)
   findlen(list,integer)


clauses

   len(X):-
      findlen(X,Count),
      write("\nLength Of List : "),
      write(Count).

   findlen([],X):-
      X=0.

   findlen([X|Tail],Count):-
      findlen(Tail,Prev),
      Count = Prev + 1.
```

Output:

```
Goal: len([a,b,c,d,e])

Length Of List : 5

Yes
```

4. *Code in prolog to find GCD of 2 Numbers.*

Code:

```
gcd(X, Y, G) :- X = Y, G = X.
gcd(X, Y, G) :-
  X < Y,
  Y1 is Y – X,
  gcd(X, Y1, G).
gcd(X, Y, G) :- X > Y, gcd(Y, X, G).
```

Output:

```
  gcd(10,5,G).
  G = 5
```

5. *Write a prolog program to check same length.*

Code:

```
same_len([],[],N):- write('Same Length').
same_len([_|T1],[_|T2],N):-
        X is N+1,
        same_len(T1,T2,X).
```

Output:

```
?- same_len([1,2,3],[4,5,6],0).
Same Length
true.
?- same_len([1,2,3],[4,5,6,7],0).
false.
?- same_len([],[],0).
Same Length
true.
```

6. *Write a prolog program to concatenate of two list.*

Code:

```
concatenation([],L,L).
concatenation([X|L1],L2,[X|L3]):-concatenation(L1,L2,L3).
```

Output:

```
concatenation([1,2],[4,5,6],X).
X = [1, 2, 4, 5, 6].

concatenation([],[4,5,6],X).
X = [4, 5, 6].
```

7. Write a prolog program to find out the maximum element of list.

Code:

```
max_l([X],X) :- !, true.
max_l([X|Xs], M):- max_l(Xs, M), M >= X.
max_l([X|Xs], X):- max_l(Xs, M), X >  M.
```

Output:

```
?- max_l([1,2,3],L).
L = 3
```

8. Write a prolog program to find out the factorial of an element.

Code:

```
factorial(0,1).
factorial(X,Y):-factorial(Z,K),X is Z+1, Y is X*K.
```

Output:

```
?- factorial(5,X).
X = 120

ˆ ?- factorial(8,X).
X = 40320
```

ASSINGMENTS-2

1. *Prolog code to find Fibonacci series.*

Code:

```
domains
   x = integer

predicates
   fibonacci(x)

clauses
   fibonacci(1).

   fibonacci(N) :-
      N1 = N - 1,
      N1 >= 0,!,
      fibonacci(N1),
      write(F1,\" ,\"),
      F = F1 + N.
```

Output:

```
6
0 1 1 2 3 5
```

2. *Prolog code to find an element is in a list or not.*

Code:

```
domains
   x = integer
   l = integer*

predicates
   find(l,x)

clauses

   find([],N) :-
      write("There is no such element in the list"),nl.

   find([Element|List],1) :-
      write("The element is ",Element),nl.

   find([Element|List],N) :-
      N1 = N-1,
      find(List,N1).
```

Output:

```
Goal: find([1,2,3,4],3)
The element is 3
Yes
```

Goal: find([1,2,3,4],0)
There is no such element in the list
Yes

Goal: find([1,2,3,4],5)
There is no such element in the list
Yes

Goal: find([1,2,4,3],4)
The element is 3
Yes

3. *Prolog code to find the reverse of the list.*

Code:

```
domains
   list=integer*
predicates
   reverse_list(list,list)
   reverse(list,list,list)
clauses

   reverse_list(Inputlist,Outputlist):-
   reverse(Inputlist,[],Outputlist).
   reverse([],Outputlist,Outputlist).

   reverse([Head|Tail],List1,List2):-
      reverse(Tail,[Head|List1],List2).
```

Output:

Goal: reverse_list([1,2,
3],X)
X=[3,2,1]
1 Solution
Goal:

4. *Prolog code to find the last elemnt of the list.*

Code:

```
  domains
   list=symbol*
predicates
   last(list)
clauses
   last([X]):-
      write("\nLast element is : "),
      write(X).

   last([Y|Tail]):-
      last(Tail).
```

Output:

Goal: last([a,b,c,d,e])
Last element is : e
Yes

ASSINGMENT-3

1. *Code in prolog to delete the occurrence of all element.*

Code:

```
domains

   list=symbol*


predicates

   del(symbol,list,list)


clauses

   del(X,[X|Tail],Tail).

   del(X,[Y|Tail],[Y|Tail1]):-
      del(X,Tail,Tail1).
```

Output:

```
Goal: del(c,[a,b,c,d,e],NewList)

NewList=["a","b","d","e"]

1 Solution

--------------------------------
Goal: del(a,[b,a,c,a],L)

L=["b","c","a"]

L=["b","a","c"]

2 Solutions
```

2. *Code in prolog to find intersection of 2 list.*

Ans.) **Code:**

```
intersectionTR(_, [], []).
intersectionTR([], _, []).
intersectionTR([H1|T1], L2, [H1|L]):-
   member(H1, L2),
   intersectionTR(T1, L2, L), !.
intersectionTR([_|T1], L2, L):-
   intersectionTR(T1, L2, L).

intersection(L1, L2):-
   intersectionTR(L1, L2, L),
   write(L).
```

Output:

```
?- intersect([1,3,5,2,4] ,[6,1,2]).
```

3. *Code in prolog to find union of 2 list.*

Ans.) **Code:**

```
nionTR([], [], []).
unionTR([], [H2|T2], [H2|L]):-
   intersectionTR(T2, L, Res),
   Res = [],
   unionTR([], T2, L),
   !.
unionTR([], [_|T2], L):-
   unionTR([], T2, L),
   !.

unionTR([H1|T1], L2, L):-
   intersectionTR([H1], L, Res),
   Res \= [],
   unionTR(T1, L2, L).
unionTR([H1|T1], L2, [H1|L]):-
   unionTR(T1, L2, L).

union(L1, L2):-
   unionTR(L1, L2, L),
   write(L).
```

Output:

```
?- union([1,3,5,2,4] ,[6,1,2]).
```

4. *Code in prolog to Divide a list in 2 equal parts.*

Ans.) **Code:**

```
domains
   list=integer*

predicates
   split(list,list,list)
clauses
   split([],[],[]).
   split([X|L],[X|L1],L2):-
      X>= 0,
      !,
      split(L,L1,L2).

   split([X|L],L1,[X|L2]):-
      split(L,L1,L2).
```

Output:

```
Goal: split([1,2,-3,4,-5
,2],X,Y)
X=[1,2,4,2], Y=[-3,-5]
1 Solution
```

ASSINGMENT-4

1. *Code in prolog to find Max of 2 numbers using CUT .*

Code:

```
predicates
go.
max(integer,integer,integer).
clauses
go:-
readint(X), readint(Y),
max(X,Y,_).
max(X,Y,X):- X>Y ,!.
max(X,Y,Y).
```

Output:

```
Goal:  max(22,33,X)
X=33
1 Solution
```

2. *Code in prolog to find sum of the list using accumulator.*

Code:

```
domains
   x = integer
   l = integer*

predicates
   sum(l,x)

clauses
   sum([],0).

   sum([X|List],Sum) :-
      sum(List,Sum1),
      Sum = X + Sum1.
```

Output:

```
Goal: sum([1,2,3,4],Sum)
Sum=10
1 Solution

Goal: sum([-2,-1,1,2],Sum)
Sum=0
1 Solution
```

Goal: sum([],Sum)
Sum=0
1 Solution

Goal: sum([1],Sum)
Sum=1
1 Solution

3. *Code in prolog to find GCD of 2 numbers using CUT.*

Code:

```
gcd(X, Y, G) :- X = Y, G = X.
gcd(X, Y, G) :-
  X < Y,
  Y1 is Y – X,
  gcd(X, Y1, G).
gcd(X, Y, G) :- X > Y, gcd(Y, X, G).
```

Output:

```
gcd(10,5,G).

 G = 5
```

ASSINGMENT-5

1. *Code in prolog to do a merge sort of the list.*

Code:

```
splitlist(L, [], L, 0).
splitlist([H|T], [H|A], B, N) :- Nminus1 is N-1, splitlist(T, A, B, Nminus1).

halfhalf(L, A, B) :- length(L, Len), Half is Len//2, splitlist(L, A, B, Half).

merge(A, [], A).
merge([], B, B).
merge([Ha|Ta], [Hb|Tb], R) :- Ha =< Hb, merge(Ta, [Hb|Tb], M), R = [Ha|M].
merge([Ha|Ta], [Hb|Tb], R) :- Ha > Hb, merge(Tb, [Ha|Ta], M), R = [Hb|M].

fkingsort([], []).
fkingsort([E], [E]).
fkingsort([H1, H2], [H1, H2]) :- H1 =< H2.
fkingsort([H1, H2], [H2, H1]) :- H1 > H2.
fkingsort(L, R) :- halfhalf(L, A, B), fuckingsort(A, Asort), fuckingsort(B, Bsort), merge(Asort,
Bsort, R).
```

Output:

```
fkingsort([1,4,5,8,6])
1,4,5,6,8
```

2. *Code in prolog to do a Quick Sort.*

Ans.)

```
quicksort([X|Xs],Ys) :-
  partition(Xs,X,Left,Right),
  quicksort(Left,Ls),
  quicksort(Right,Rs),
  append(Ls,[X|Rs],Ys).
quicksort([],[]).

partition([X|Xs],Y,[X|Ls],Rs) :-
  X <= Y, partition(Xs,Y,Ls,Rs).
partition([X|Xs],Y,Ls,[X|Rs]) :-
  X > Y, partition(Xs,Y,Ls,Rs).
partition([],Y,[],[]).

append([],Ys,Ys).
append([X|Xs],Ys,[X|Zs]) :- append(Xs,Ys,Zs).
```

Output:

```
quicksort([1,4,5,8,6])
1,4,5,6,8
```

3. *Code in prolog to do a insertion sort.*

Code:

```
insert(X,[],[X]).
insert(X,[H|T],[X,H|T]):- X =< H,!.
insert(X,[H|T1],[H|T2]):- insert(X,T1,T2).
```

```
insort([],[]).
insort([H|T],Sorted):- insort(T,Sorted2), insert(H,Sorted2,Sorted).
```

Output:

```
insert([1,4,5,8,6])
1,4,5,6,8
```

4. *Code in prolog to do a selection sort.*

Code:

```
selectionsort([],[]).
selectionsort([A],[A]).
selectionsort(In,[Min|Out]):-
  findMin(In,Min),
  removeElement(In,Min,L),
  selectionsort(L,Out).

findMin([A]|[A]).
findMin([H|T], H):-
  findMin(T,Z),
  H <= Z.
findMin([H|T],Z):-
  findMin(T,Z),
  H > Z.

removeElement([A],A,[]).
removeElement([H|T],H,T).
removeElement([H|T],X,[H|Z]):-
  X /== H,
  removeElement(T,X,Z).
```

Output:

```
selectionsort([1,4,5,8,6])
1,4,5,6,8
```

5. *Code in prolog to do a bubble sort.*

Ans.)**Code:**

```
bubblesort([],[]).
bubblesort(In, [H|Out]):-
  pulldown(In, [H|T]),
  bubblesort(T, Out),

pulldown([],[]).
pulldown([A], [A]).
pulldown([H|T], [H,X|Y]):-
  pulldown(T,[X|Y]),
  H <= X.

pulldown([H|T], [X,H|Y]):-
  pulldown(T,[X|Y]),
  H > X.
```

Output:

```
bubblesort([1,4,5,8,6])
1,4,5,6,8
```

CONCLUSION

Prolog is a logic programming language associated with artificial intelligence and computational linguistics.

It has given us a lot of knowledge regarding the working of Artificial Intelligence Systems and their ability understand and interpret knowledge and draw plausible conclusions in the form of either yes or no. This lab has has really helped us in developing our skill so that we can opt for this sector of industry if we want to.