Artificial Intelligence
Practical #1

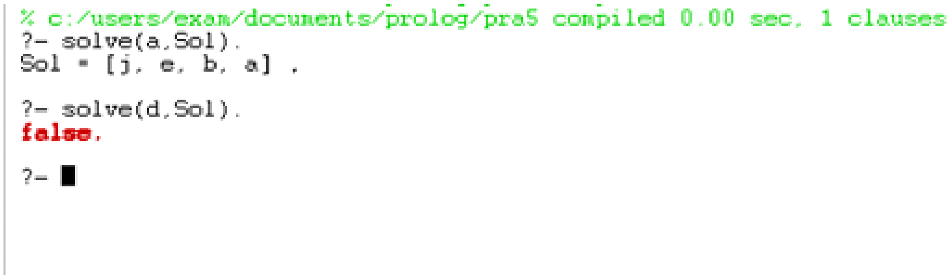
Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic	**Search Algorithm**		

1. Write a program to implement depth first search algorithm.

Code:

```
solve(Node , Solution):-depthfirst([],Node,Solution).
depthfirst(Path,Node,[Node | Path]):-goal(Node).
depthfirst(Path,Node,Sol):-
   s(Node,Node1),
   \+member(Node1,Path), % Prevent a cycle
   depthfirst([Node | Path],Node1,Sol).
   % Data for Graph
s(a,b).
s(a,c).
s(b,d).
s(b,e).
s(c,f).
s(c,g).
s(d,h).
s(e,i).
s(e,j).
goal(j).
goal(f).
```

O/P:-



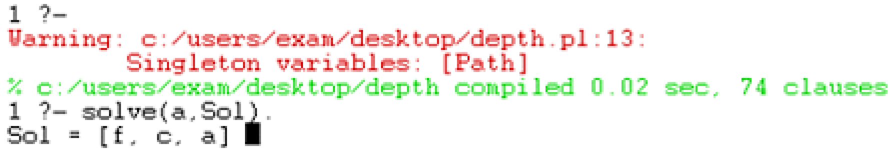
```
% c:/users/exam/documents/prolog/pra5 compiled 0.00 sec, 1 clauses
?- solve(a,Sol).
Sol = [j, e, b, a] .

?- solve(d,Sol).
false.

?- 
```

2. Write a program to implement breadth first search algorithm.

Code:

```
solve(Start,Solution):-
    breadthfirst([[Start]],Solution).
 breadthfirst([[Node|Path]|_],[Node|Path]):-goal(Node).
 breadthfirst([Path|Paths],Solution):-
    extend(Path,NewPaths),
    append(Paths,NewPaths,Paths1),
    breadthfirst(Paths1,Solution).
  extend([Node|Path],NewPaths):-
    bagof([NewNode,Node|Path],
    (s(Node,NewNode),\+member(NewNode,[Node|Path])),
    NewPaths),
    !.
  extend(Path,[] ).
s(a,b).
  s(a,c).
  s(b,d).
  s(b,e).
  s(c,f).
  s(c,g).
  s(d,h).
  s(e,i).
  s(e,j).
  goal(j).
  goal(f).
```

O/P:-



```
1 ?-
Warning: c:/users/exam/desktop/depth.pl:13:
        Singleton variables: [Path]
% c:/users/exam/desktop/depth compiled 0.02 sec, 74 clauses
1 ?- solve(a,Sol).
Sol = [f, c, a] █
```

Artificial Intelligence
Practical #2

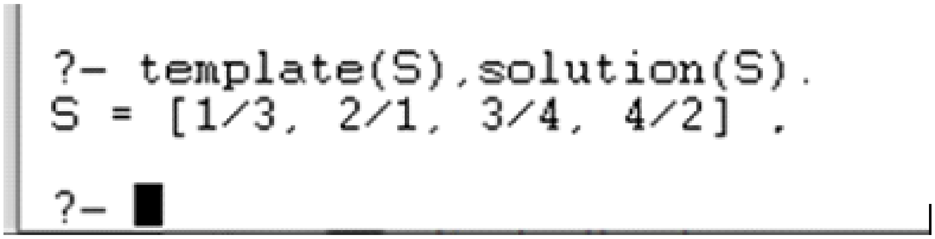
Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic			

1. Write a program to simulate 4-Queen / N-Queen problem.

Code:

```
solution([]).
  solution([X/Y|Others]):-
     solution(Others),member(Y,[1,2,3,4]),noattack(X/Y,Others).

noattack(_,[]).
  noattack(X/Y,[X1/Y1|Others]):-Y=\=Y1,Y1-Y=\=X1-X,Y1-Y=\=X-X1,noattack(X/Y,Others).
  template([1/Y1,2/Y2,3/Y3,4/Y4]).
```

Output:



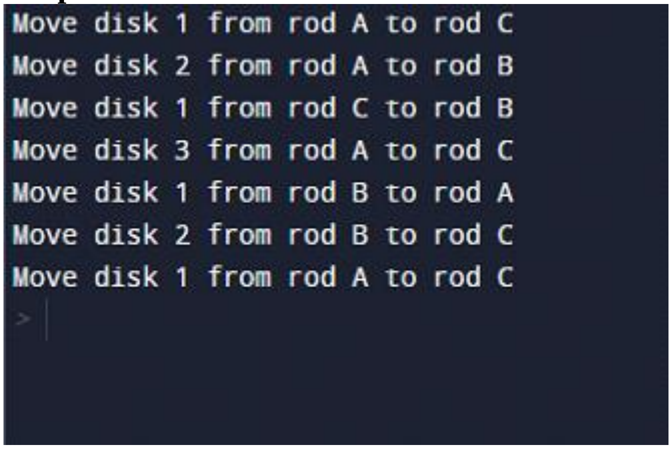
```
?- template(S),solution(S).
S = [1/3, 2/1, 3/4, 4/2] .

?- 
```

2. Write a program to solve tower of Hanoi problem.

Code:

```
def TowerOfHanoi(n, from_rod, to_rod, aux_rod):
if n == 0:
return
TowerOfHanoi(n-1, from_rod, aux_rod, to_rod)
print("Move disk", n, "from rod", from_rod, "to rod", to_rod)
TowerOfHanoi(n-1, aux_rod, to_rod, from_rod)
N = 3
TowerOfHanoi(N, 'A', 'C', 'B')
```

Output:



```
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
>
```

3. Write a program to solve the Monkey Banana problem.

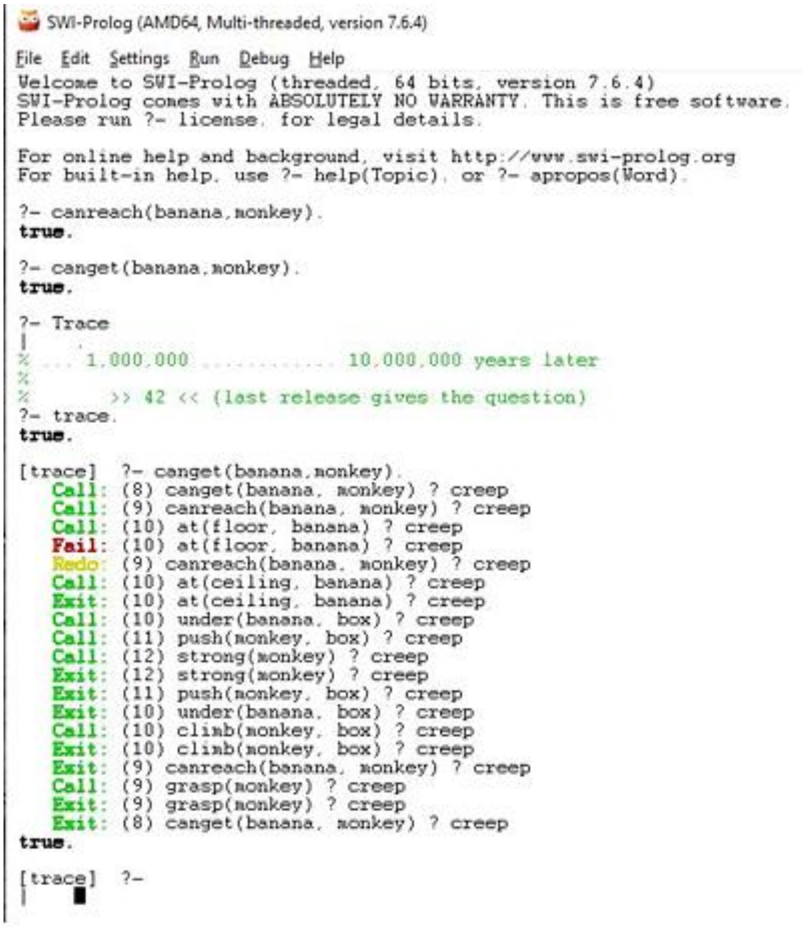
Code:

```
on(floor,monkey).
on(floor,box).
in(room,monkey).
in(room,box).
in(room,banana).
at(ceiling,banana).
strong(monkey).
grasp(monkey).
climb(monkey,box).
push(monkey,box):-strong(monkey).
under(banana,box):-push(monkey,box).
canreach(banana,monkey):-
   at(floor,banana);
   at(ceiling,banana),
   under(banana,box),
   climb(monkey,box).
canget(banana,monkey):-
   canreach(banana,monkey),
   grasp(monkey).
```

Or

```
move(state(middle,onbox,middle,hasnot),
  grasp,
  state(middle,onbox,middle,has)).
move(state(P,onfloor,P,H),
  climb,
  state(P,onbox,P,H)).
move(state(P1,onfloor,P1,H),
  drag(P1,P2),
  state(P2,onfloor,P2,H)).
move(state(P1,onfloor,B,H),
  walk(P1,P2),
  state(P2,onfloor,B,H)).
canget(state(_,_,_,has)).
canget(State1) :-
  move(State1,_,State2),
  canget(State2).
```

Output:


SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- canreach(banana,monkey).
true.

?- canget(banana,monkey).
true.

?- Trace
|
% ... 1,000,000 ............ 10,000,000 years later
%
%       >> 42 << (last release gives the question)
?- trace.
true.

[trace]  ?- canget(banana,monkey).
   Call: (8) canget(banana, monkey) ? creep
   Call: (9) canreach(banana, monkey) ? creep
   Call: (10) at(floor, banana) ? creep
   Fail: (10) at(floor, banana) ? creep
   Redo: (9) canreach(banana, monkey) ? creep
   Call: (10) at(ceiling, banana) ? creep
   Exit: (10) at(ceiling, banana) ? creep
   Call: (10) under(banana, box) ? creep
   Call: (11) push(monkey, box) ? creep
   Call: (12) strong(monkey) ? creep
   Exit: (12) strong(monkey) ? creep
   Exit: (11) push(monkey, box) ? creep
   Exit: (10) under(banana, box) ? creep
   Call: (10) climb(monkey, box) ? creep
   Exit: (10) climb(monkey, box) ? creep
   Exit: (9) canreach(banana, monkey) ? creep
   Call: (9) grasp(monkey) ? creep
   Exit: (9) grasp(monkey) ? creep
   Exit: (8) canget(banana, monkey) ? creep
true.

[trace]  ?-
|
```

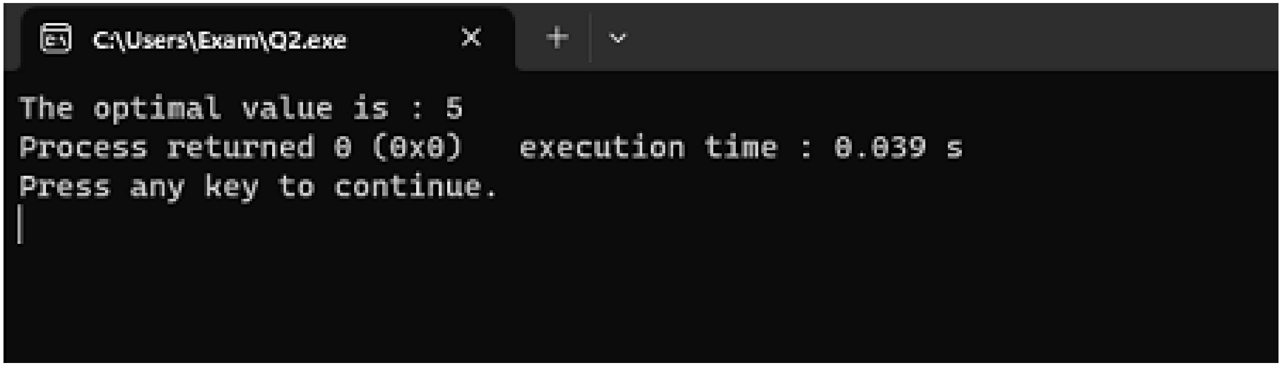
Artificial Intelligence
Practical #3

Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic	**Search Algorithm**		

1. Write a program to implement alpha beta search.

```
// C++ program to demonstrate
// working of Alpha-Beta Pruning
#include <bits/stdc++.h>
using namespace std;
// Initial values of
// Alpha and Beta
const int MAX = 1000;
const int MIN = -1000;
// Returns optimal value for
// current player(Initially called
// for root and maximizer)
int minimax(int depth, int nodeIndex,
      bool maximizingPlayer,
      int values[], int alpha,
      int beta)
{

  // Terminating condition. i.e
  // leaf node is reached
  if (depth == 3)
    return values[nodeIndex];

  if (maximizingPlayer)
  {
    int best = MIN;

    // Recur for left and
    // right children
    for (int i = 0; i < 2; i++)
    {

      int val = minimax(depth + 1, nodeIndex * 2 + i,
               false, values, alpha, beta);
      best = max(best, val);
      alpha = max(alpha, best);

      // Alpha Beta Pruning
      if (beta <= alpha)
        break;
    }
    return best;
  }
  else
  {
    int best = MAX;

    // Recur for left and
    // right children
    for (int i = 0; i < 2; i++)
    {
      int val = minimax(depth + 1, nodeIndex * 2 + i,
               true, values, alpha, beta);
      best = min(best, val);
      beta = min(beta, best);

      // Alpha Beta Pruning
      if (beta <= alpha)
        break;
    }
    return best;
  }
}

// Driver Code
int main()
{
  int values[8] = {3, 5, 6, 9, 1, 2, 0, -1};
  cout << "The optimal value is : " << minimax(0, 0, true, values, MIN, MAX);
  ;
  return 0;
}
```

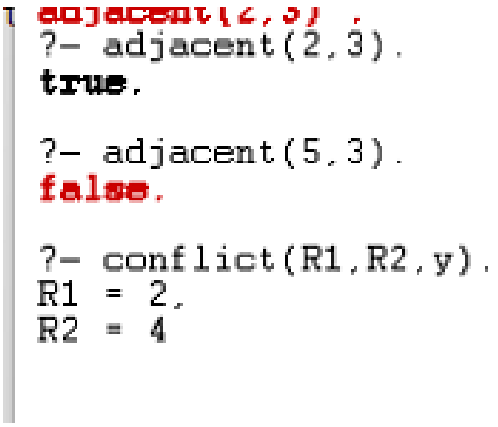
O/P:-



```
The optimal value is : 5
Process returned 0 (0x0)   execution time : 0.039 s
Press any key to continue.
```

2. Write a program for Hill climbing problem.

```
adjacent(1,2).
adjacent(1,3).
adjacent(1,4).
adjacent(1,5).
adjacent(2,3).
adjacent(2,4).
adjacent(3,4).
adjacent(4,5).
adjacent(2,1).
adjacent(3,1).
adjacent(4,1).
adjacent(5,1).
adjacent(3,2).
adjacent(4,2).
adjacent(4,3).
adjacent(5,4).
color(1,orange,x).
color(1,orange,y).
color(2,pink,x).
color(2,pink,y).
color(3,purple,x).
color(3,purple,y).
color(4,red,x).
color(4,pink,y)
color(5,pink,x).
color(5,purple,y).
conflict(Coloring):- adjacent(A,B),
    color(A,Color,Coloring),
    color(B,Color,Coloring).
conflict(R1,R2,Coloring):- adjacent(R1,R2),
    color(R1,Color,Coloring),
    color(R2,Color,Coloring).
```

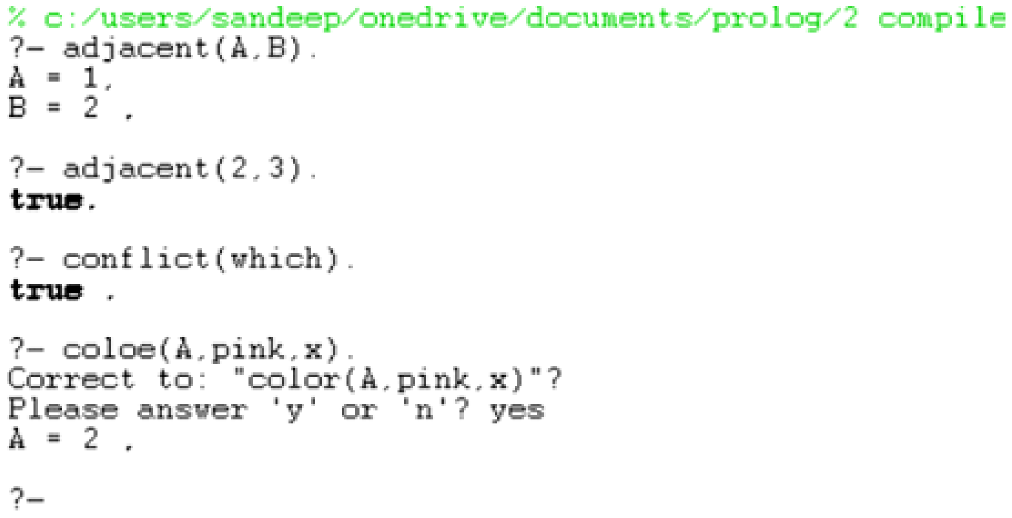


```
adjacent(2,3) .
?- adjacent(2,3).
true.

?- adjacent(5,3).
false.

?- conflict(R1,R2,y).
R1 = 2,
R2 = 4
```

2CODE:-

```
adjacent(1,2).
adjacent(1,3).
adjacent(1,4).
adjacent(1,5).
adjacent(2,3).
adjacent(2,4).a
adjacent(3,4).
adjacent(4,5).
adjacent(2,1).
adjacent(3,1).
adjacent(4,1).
adjacent(5,1).
adjacent(3,2).
adjacent(4,2).
adjacent(4,3).
adjacent(5,4).
color(1,orange,x).
color(1,orange,y)
color(2,pink,x).
color(2,pink,y).
color(3,purple,x).
color(3,purple,y).
color(4,red,x).
color(4,pink,y).
color(5,pink,x).
color(5,purple,y).
conflict(Coloring):- adjacent(A,B),
    color(A,Color,Coloring),
    color(B,Color,Coloring).
conflict(R1,R2,Coloring):- adjacent(R1,R2),
    color(R1,Color,Coloring),
    color(R2,Color,Coloring).
```

O/P:-



```
% c:/users/sandeep/onedrive/documents/prolog/2 compile
?- adjacent(A,B).
A = 1,
B = 2 .

?- adjacent(2,3).
true.

?- conflict(which).
true .

?- coloe(A,pink,x).
Correct to: "color(A,pink,x)"?
Please answer 'y' or 'n'? yes
A = 2 .

?-
```

VSIT

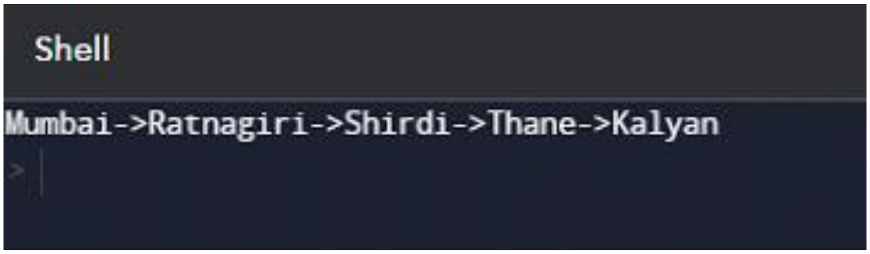
Artificial Intelligence
Practical #4

Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic	**Search Algorithm**		

1. Write a program to implement A* algorithm.

```
graph={
    'Mumbai':set(['Nagpur', 'Ratnagiri', 'Solapur']),
    'Solapur' :set(['Nashik', 'Mumbai']),
    'Nashik':set(['Solapur', 'Ratnagiri']),
    'Nagpur':set(['Mumbai', 'Ahmednagar']),
    'Ahmednagar':set(['Nagpur', 'Murud']),
    'Murud':set(['Ahmednagar', 'Pune']),
    'Pune':set(['Murud', 'Panvel']),
    'Panvel':set(['Pune', 'Shirdi', 'Thane']),
    'Thane':set(['Panvel', 'Shirdi', 'Kalyan']),
    'Shirdi':set(['Panvel', 'Thane', 'Ratnagiri']),
    'Ratnagiri':set([ 'Mumbai', 'Nashik', 'Shirdi', 'Satara']),
    'Satara':set(['Ratnagiri', 'Kalyan']),
    'Kalyan':set(['Satara', 'Thane', 'Giurgiu'])
}



pc={
    ('Mumbai','Solapur'):75,
    ('Mumbai','Ratnagiri'):140,
    ('Mumbai','Nagpur'):118,
    ('Solapur','Mumbai'):75,
    ('Solapur','Nashik'):71,
    ('Nashik','Ratnagiri'):151,
    ('Nashik','Solapur'):71,
    ('Nagpur','Ahmednagar'):111,
    ('Nagpur','Mumbai'):118,
    ('Ahmednagar','Murud'):70,
    ('Ahmednagar','Nagpur'):111,
    ('Murud','Pune'):75,
    ('Murud','Ahmednagar'):70,
    ('Pune','Panvel'):120,
    ('Pune','Murud'):70,
    ('Panvel','Thane'):138,
    ('Panvel','Shirdi'):146,
    ('Panvel','Pune'):120,
    ('Thane','Kalyan'):101,
    ('Thane','Shirdi'):97,
    ('Thane','Panvel'):198,
    ('Shirdi','Sibia'):80,
    ('Shirdi','Thane'):97,
    ('Shirdi','Panvel'):146,
    ('Ratnagiri','Panvel'):77,
    ('Ratnagiri','Nashik'):151,
    ('Ratnagiri', 'Mumbai'):140,
    ('Ratnagiri', 'Shirdi'):80,
    ('Ratnagiri', 'Satara'):99,
    ('Satara', 'Kalyan'):211,
    ('Satara', 'Ratnagiri'):99,
    ('Kalyan', 'Thane'):101,
    ('Kalyan', 'Satara'):211
}



h={
    'Mumbai':366,
    'Kalyan': 0,
    'Panvel':160,
    'Pune': 242,
    'Ahmednagar':244,
    'Nashik': 380,
    'Thane':100,
    'Solapur':374,
    'Ratnagiri':253,
    'Nagpur' : 329,
    'Murud': 241,
    'Shirdi':193,
    'Satara':176
}



def dfs(g, v, goal, explored, path,m):
    explored.add(v)
    node={}
    if v==goal:
        return path+v
    for w in g[v]: #w is next node
        if w not in explored:
            f=h.get(w)+pc[v,w]
            if m>f:
                m=f
                node=w
    P=dfs(g, node, goal, explored, path + v + '->', m)
    if P:
        return P
print(dfs(graph, 'Mumbai', 'Kalyan',set(),' ', 1000))
```

O/P:-



```
Mumbai->Ratnagiri->Shirdi->Thane->Kalyan
```

Artificial Intelligence
Practical #5

Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic			

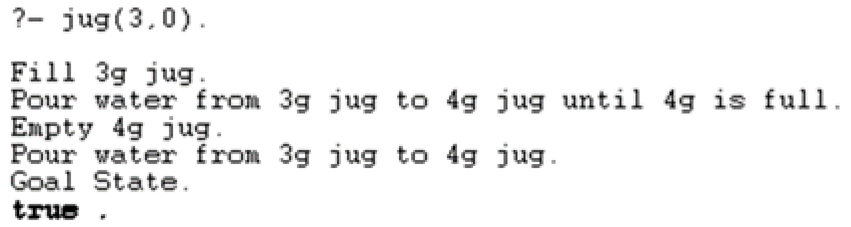
1. Write a program to solve water jug problem.

Code:

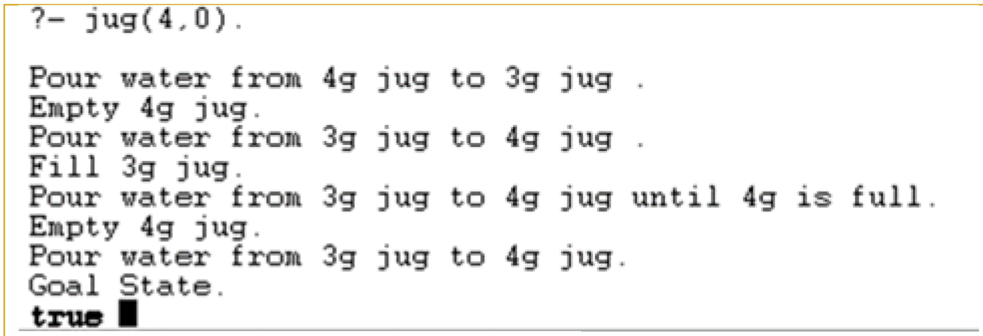
CODE:

```
jug(0,0):-write("\nFill 3g jug."),jug(0,3).
   jug(0,3):-write("\nPour water from 3g jug to 4g jug ."),jug(3,0).
   jug(3,0):-write("\nFill 3g jug."),jug(3,3).
   jug(3,3):-write("\nPour water from 3g jug to 4g jug until 4g is full."),jug(4,2).
   jug(4,2):-write("\nEmpty 4g jug."),jug(0,2).
   jug(0,2):-write("\nPour water from 3g jug to 4g jug."),jug(2,0).
   jug(2,0):-write("\nGoal State.").

jug(X,Y):-X>4,not(Y>3),write("\n4g jug overflowed."),not(jug(2,0)).
   jug(X,Y):-notX>4,(Y>3),write("\n3g jug overflowed."),not(jug(2,0)).
   jug(X,Y):-X>4,(Y>3),write("\nBoth jug overflowed."),not(jug(2,0)).
jug(0,0):-write("\nFill 4g jug."),jug(4,0).
   jug(4,0):-write("\nPour water from 4g jug to 3g jug ."),jug(1,3).
   jug(1,3):-write("\nEmpty 4g jug."),jug(0,3).
   jug(0,3):-write("\nPour water from 3g jug to 4g jug until 3g is full."),jug(3,0).
   jug(3,0):-write("\nFill 3g jug."),jug(3,3).
   jug(3,3):-write("\nPour water from 3g jug to 4g jug until 4g is full."),jug(4,2).
   jug(4,2):-write("\nEmpty 4g jug."),jug(0,2).
   jug(0,2):-write ("\nPour water from 3g jug to 4g until 3g id Empty jug."),jug(2,0).
   jug(2,0):-write ("\nGoal State.").
```

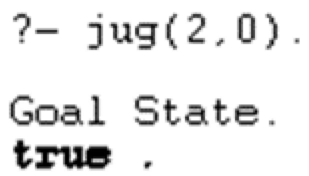
OUTPUT:



```
?- jug(3,0).

Fill 3g jug.
Pour water from 3g jug to 4g jug until 4g is full.
Empty 4g jug.
Pour water from 3g jug to 4g jug.
Goal State.
true .
```



```
?- jug(4,0).

Pour water from 4g jug to 3g jug .
Empty 4g jug.
Pour water from 3g jug to 4g jug .
Fill 3g jug.
Pour water from 3g jug to 4g jug until 4g is full.
Empty 4g jug.
Pour water from 3g jug to 4g jug.
Goal State.
true
```



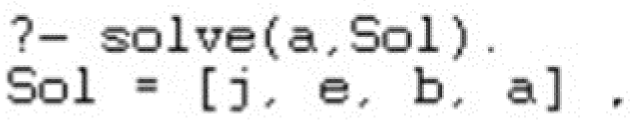
```
?- jug(2,0).

Goal State.
true .
```

2. Design the simulation of tic – tac – toe game using min-max algorithm.

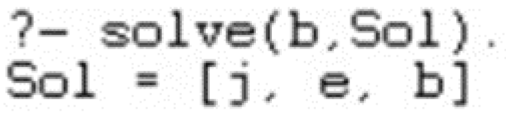
CODE:

```
solve(Node,Solution):-depthfirst([],Node,Solution).
depthfirst(Path,Node,[Node|Path]):- goal(Node).
depthfirst(Path,Node,Sol):-
   s(Node,Nodel),
   \+member(Nodel,Path),
   depthfirst([Node|Path],Nodel,Sol).
s(a,b).
s(a,c).
s(b,d).
s(b,e).
s(c,f).
s(d,h).
s(e,i).
s(e,j).
goal(j).
goal(f).
```

OUTPUT:



```
?- solve(a,Sol).
Sol = [j, e, b, a] .
```



```
?- solve(b,Sol).
Sol = [j, e, b]
```

VSIT

Artificial Intelligence
Practical #7

Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic			

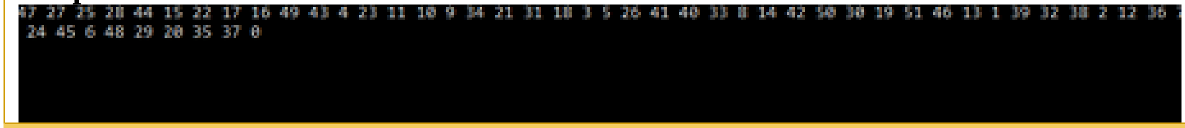
1. Write a program to shuffle Deck of cards.

Code:

Code in Visual Studio 2017:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp5
{
    class Program
    {
        public static void shuffle(int []a)
        {
            Random rand = new Random();
            for(int i=0;i<=51;i++)
            {
               int r = rand.Next(52); //0-51
                int temp = a[i];
                a[i] = a[r];
                a[r] = temp;
            }
        }
      static void Main(string[] args)
        {
            int[] a = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51 };
            shuffle(a);
            foreach(int x in a)
            {
                Console.Write(x + " ");

            }
            Console.ReadLine();
        }
    }
}
```

Output:



```
47 27 25 28 44 15 22 17 16 49 43 4 23 11 10 9 34 21 31 18 3 5 26 41 40 33 8 14 42 50 30 19 51 46 13 1 39 32 38 2 12 36
 24 45 6 48 29 20 35 37 0
```

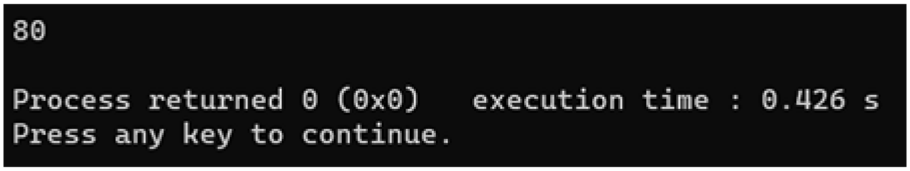
2.Solve traveling salesman problem using artificial intelligence technique.

Code:

Code in C++

```
#include<bits/stdc++.h>
using namespace std;
#define V 4
int travllingSalesmanProblem(int graph[][V],int s)
{
   vector <int>vertex;
   for(int i=0;i<V;i++)
      if(i!=s)
         vertex.push_back(i);

   int min_path=INT_MAX;
   do{
      int current_pathweigth=0;
      int k=s;
      for (int i=0;i<vertex.size();i++)
         {
         current_pathweigth+=graph[k][vertex[i]];
         k=vertex[i];
      }
current_pathweigth+=graph[k][s];
   min_path=min(min_path,current_pathweigth);
   }while(
        next_permutation(vertex.begin(),vertex.end()));
         return min_path;
}
int main()
{

   int graph[][V]={{0,10,15,20},{10,0,35,25},{15,35,0,30},{20,25,30,0}};
   int s=0;
   cout<<travllingSalesmanProblem(graph,s)<<endl;
   return 0;
}
```

Output:

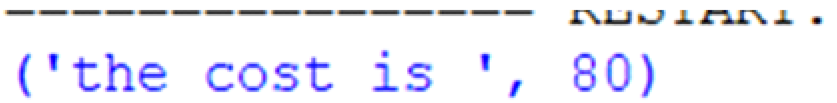


```
80

Process returned 0 (0x0)   execution time : 0.426 s
Press any key to continue.
```

Code in Python:

```
from sys import maxsize
from itertools import permutations
V = 4
def travllingSalesmanProblem(graph, s):

    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)
    min_path = maxsize
    next_permutation = permutations(vertex)
    for i in next_permutation:
        current_pathweight = 0
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]
        min_path = min(min_path, current_pathweight)
    return ("the cost is " , min_path)
if __name__ == "__main__":
    graph = [[0, 10, 15, 20], [10, 0, 35, 25], [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print(travllingSalesmanProblem(graph, s))
```

O/P:-



```
================ RESTART:
('the cost is ', 80)
```

Artificial Intelligence
Practical #8

Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic	**Search Algorithm**		

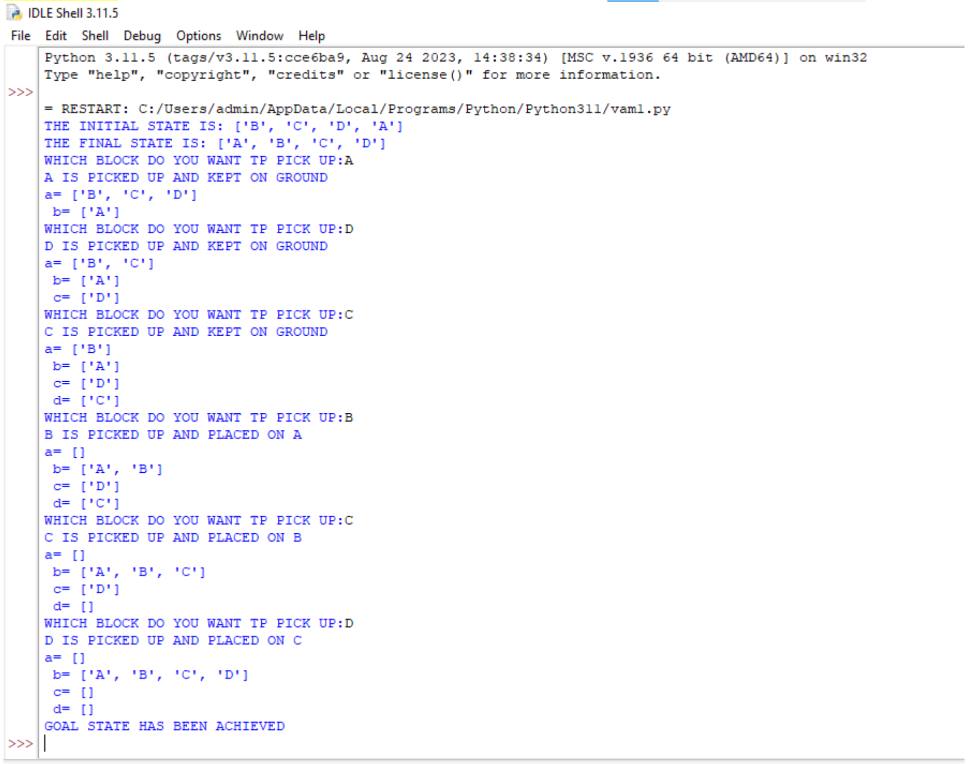
1. Solve the block of World problem.

```
a=["B","C","D","A"]
k=['A','B','C','D']
print('THE INITIAL STATE IS:',a)
print('THE FINAL STATE IS:',k)
b=[]
c=[]
d=[]
while True:
add=str(input("WHICH BLOCK DO YOU WANT TP PICK UP:"))
if(add=='A'):
print('A IS PICKED UP AND KEPT ON GROUND')
b.append(add)
a.remove(add)
print("a=",a,"\n","b=",b)
add=input("WHICH BLOCK DO YOU WANT TP PICK UP:")
if(add=='D'):
print('D IS PICKED UP AND KEPT ON GROUND')
c.append(add)
a.remove(add)
print("a=",a,'\n',"b=",b,'\n',"c=",c)
add=input("WHICH BLOCK DO YOU WANT TP PICK UP:")
if(add=='C'):
print('C IS PICKED UP AND KEPT ON GROUND')
d.append(add)
a.remove(add)
print("a=",a,'\n',"b=",b,'\n',"c=",c,'\n',"d=",d)
add=input("WHICH BLOCK DO YOU WANT TP PICK UP:")
if(add=='B'):
print('B IS PICKED UP AND PLACED ON A')
b.append(add)
a.remove(add)
print("a=",a,'\n',"b=",b,'\n',"c=",c,'\n',"d=",d)
add=input("WHICH BLOCK DO YOU WANT TP PICK UP:")
if add=='C':
print('C IS PICKED UP AND PLACED ON B')
b.append(add)
d.remove(add)
print("a=",a,'\n',"b=",b,'\n',"c=",c,'\n',"d=",d)
add=input("WHICH BLOCK DO YOU WANT TP PICK UP:")
```

```
if add=='D':
print('D IS PICKED UP AND PLACED ON C')
b.append(add)
c.remove(add)
if k==b:
print("a=",a,'\n',"b=",b,'\n',"c=",c,'\n',"d=",d)
print('GOAL STATE HAS BEEN ACHIEVED')
break
elif add=='A'or'B'or'C':
print('ALREADY PICKED UP \n START OVER')
break
else:
print('WRONG INPUT IS GIVEN \n START OVER')
elif add=='A'or'B':
print('ALREADY PICKED UP \n START OVER')
break
elif add=='D':
print('BLOCKS SHOULD BE PICKED UP IN ORDER \n START OVER')
break
else:
print('WRONG INPUT IS GIVEN \n START OVER')
elif add=='C' or 'D':
print('BLOCKS SHOULD BE PICKED UP IN ORDER \n PICK B NEXT TIME')
break
elif add=='A':
print ('TO ACHIEVE GOAL STATE DONT PICK UP "A" PICK THE BLOCKS IN ORDER')
break
else:
print ('WRONG INPUT IS GIVEN \n START OVER')
break
elif add=='A'or'D':
print ('ALREDY PICKED UP \n START OVER')
break
elif add=='B':
print ('BLOCKS SHOULD BE PICKED UP IN ORDER \n START OVER')
break
else:
print ('WRONG INPUT IS GIVEN \n START OVER')
break
elif add=='B'or'C':
print ('BLOCKS SHOULD BE PICKED UP IN ORDER \n START OVER')
break
elif add=='A':
print ('A ALREADY PICKED UP \n START OVER')
break
else:
print('WRONG INPUT IS GIVEN \n START OVER')
break
elif add=='B'or'C'or'D':
print ('BLOCKS SHOULD BE PICKED UP IN ORDER \n START OVER')
break
```

```
else:
print ('WRONG INPUT IS GIVEN \n START OVER')
break
```

Output:-



```
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python311/vaml.py
THE INITIAL STATE IS: ['B', 'C', 'D', 'A']
THE FINAL STATE IS: ['A', 'B', 'C', 'D']
WHICH BLOCK DO YOU WANT TP PICK UP:A
A IS PICKED UP AND KEPT ON GROUND
a= ['B', 'C', 'D']
 b= ['A']
WHICH BLOCK DO YOU WANT TP PICK UP:D
D IS PICKED UP AND KEPT ON GROUND
a= ['B', 'C']
 b= ['A']
 c= ['D']
WHICH BLOCK DO YOU WANT TP PICK UP:C
C IS PICKED UP AND KEPT ON GROUND
a= ['B']
 b= ['A']
 c= ['D']
 d= ['C']
WHICH BLOCK DO YOU WANT TP PICK UP:B
B IS PICKED UP AND PLACED ON A
a= []
 b= ['A', 'B']
 c= ['D']
 d= ['C']
WHICH BLOCK DO YOU WANT TP PICK UP:C
C IS PICKED UP AND PLACED ON B
a= []
 b= ['A', 'B', 'C']
 c= ['D']
 d= []
WHICH BLOCK DO YOU WANT TP PICK UP:D
D IS PICKED UP AND PLACED ON C
a= []
 b= ['A', 'B', 'C', 'D']
 c= []
 d= []
GOAL STATE HAS BEEN ACHIEVED
>>> 
```

VSIT

Artificial Intelligence
Practical #9

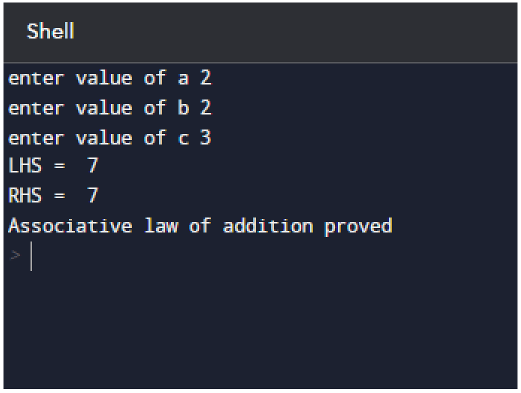
Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic			

1. Derive the expressions based on Associative law

Code :

```
a=int(input("enter value of a "));
b=int(input("enter value of b "));
c=int(input("enter value of c "));
lhs=(a+b)+c;
rhs=a+(b+c);
print("LHS = ",lhs);
print("RHS = ",rhs);
if(lhs==rhs):
    print("Associative law of addition proved");
```

Output :



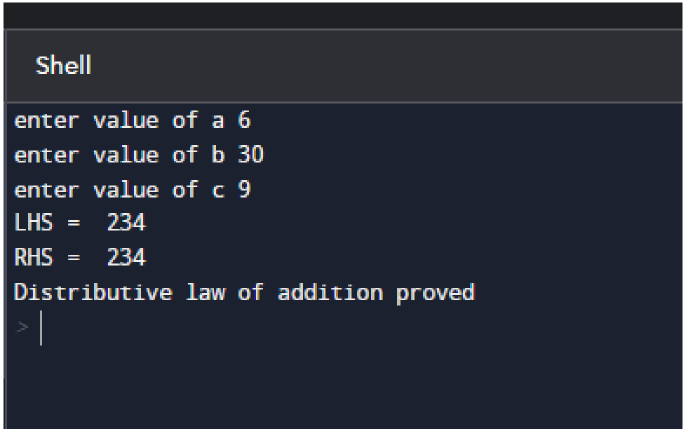
```
enter value of a 2
enter value of b 2
enter value of c 3
LHS =  7
RHS =  7
Associative law of addition proved
>
```

2. Derive the expressions based on Distributive law

Code :

```
a=int(input("enter value of a "));
b=int(input("enter value of b "));
c=int(input("enter value of c "));
lhs = a*(b+c);
rhs = ((a*b)+(a*c));
print("LHS = ",lhs);
print("RHS = ",rhs);
if(lhs==rhs):
   print("Distributive law of addition proved");
```

Output :



```
enter value of a 6
enter value of b 30
enter value of c 9
LHS =  234
RHS =  234
Distributive law of addition proved
>
```

Artificial Intelligence
Practical #10

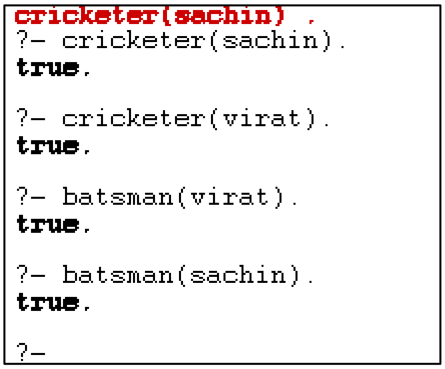
Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic	**Basics of prolog**		

1. Write a program to derive the predicate. (for e.g.: Sachin is batsman , batsman is cricketer) - > Sachin is Cricketer.

Code: -

```
batsman(sachin).
batsman(virat).
batsman(abd).
cricketer(X):-batsman(X).
```

Output: -



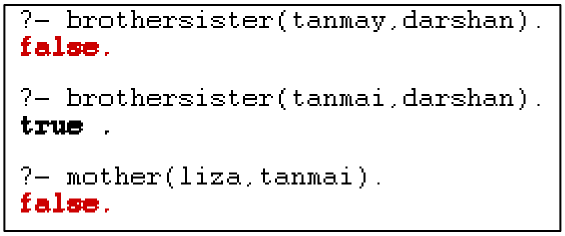
```
cricketer(sachin) .
?- cricketer(sachin).
true.

?- cricketer(virat).
true.

?- batsman(virat).
true.

?- batsman(sachin).
true.

?-
```

2. Write a program which contains three predicates: male, female, parent. Make rules for following family relations: father, mother, grandfather, grandmother, brother, sister, uncle, aunt, nephew and niece, cousin.
Question:
i. Draw Family Tree.
ii. Define: Clauses, Facts, Predicates and Rules with conjunction and disjunction

Code : -

```
Parent(Sonu,Pratik).
parent(tanmai,rohit).
parent(sarvesh,rohit).
parent(savitri,tanmai).
parent(ramesh,tanmai).
parent(savitri,darshan).
parent(ramesh,darshan).
parent(savitri,mavshi).
parent(ramesh,mavshi).
mother(X,Y):-parent(X,Y),female(X).
father(X,Y):-parent(X,Y),male(X).
haschild(X):-parent(X,_).
sister(X,Y):-parent(Z,X),parent(Z,Y),female(X),X\==Y.
brother(X,Y):-parent(Z,X),parent(Z,Y),male(X),X\==Y.
grandfather(X,Y):-parent(X,Z),parent(Z,Y),male(X).
grandmother(X,Y):-parent(X,Z),parent(Z,Y),female(X).
brothersister(X,Y):-parent(Z,X),parent(Z,Y),female(X),male(Y).
```

Output : -



```
?- brothersister(tanmay,darshan).
false.

?- brothersister(tanmai,darshan).
true .

?- mother(liza,tanmai).
false.
```

VSIT

Artificial Intelligence
Practical #11

Name	Sonu Kahar	**Roll Number**	21302A0053
Subject/Course:	**Artificial Intelligence**		
Topic	**Basics of prolog**		

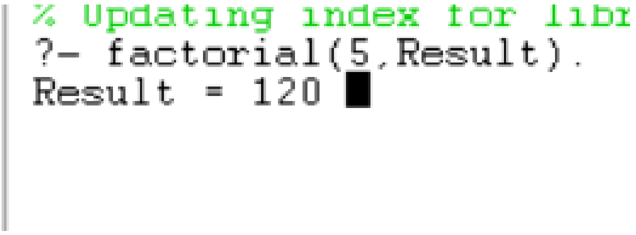
Write a prolog program to find out the factorial of a number.
Write a prolog program to compute Fibonacci series.
Write a Prolog Program to add two numbers.
Write a prolog program to find out whether input number is odd or even
Add an element in front of the list.
To delete an element from a list
Find out the sublist from a list.
Write a Prolog Program to find out the largest number from a list.

1) **Program to find out the Factorial of Number:-**

Code:

```
factorial(0, 1).

factorial(N, Result) :-

   N > 0,

   N1 is N - 1,

   factorial(N1, SubResult),

   Result is N * SubResult.
```

Output:



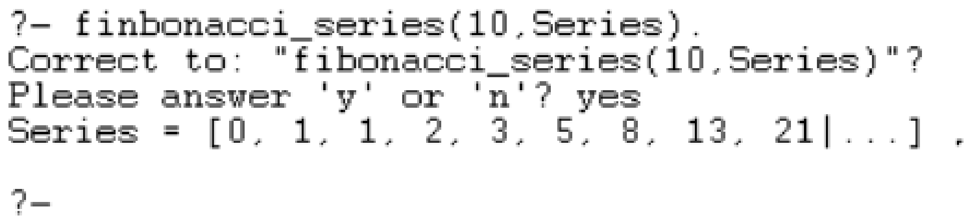
```
% Updating index for libr
?- factorial(5,Result).
Result = 120
```

2) Write a Prolog Code to Compute Fibonacci Series.

Code:

```
fibonacci_series(0, [0]).
fibonacci_series(1, [0, 1]).

% Calculate the Fibonacci series as a list up to N
fibonacci_series(N, Series) :-
    N > 1,
    N1 is N - 1,
    N2 is N - 2,
    fibonacci_series(N1, Series1),
    nth0(N1, Series1, Prev1),
    nth0(N2, Series1, Prev2),
    Next is Prev1 + Prev2,
    append(Series1, [Next], Series).
```

Output:



```
?- finbonacci_series(10,Series).
Correct to: "fibonacci_series(10,Series)"?
Please answer 'y' or 'n'? yes
Series = [0, 1, 1, 2, 3, 5, 8, 13, 21|...] .

?-
```

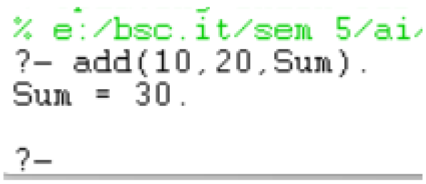
---

3) Write a Prolog Code to add Two Numbers

Code:

```
Add(X,Y,Sum):-

    Sum is X+Y.
```

Output:



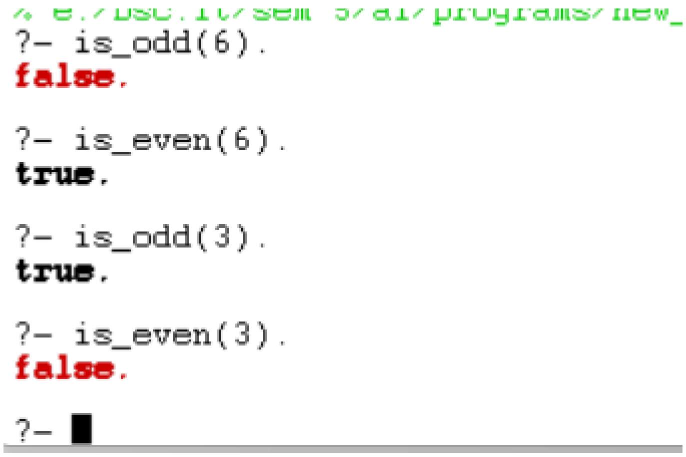
```
% e:/bsc.it/sem 5/ai/
?- add(10,20,Sum).
Sum = 30.

?-
```

4) Code to find if the given number is odd or even.

Code:

```
is_even(N) :-
    N mod 2 =:= 0.
is_odd(N) :-
    N mod 2 =:= 1.
```

Output:



```
% e:/bsc.it/sem 5/ai/programs/new_
?- is_odd(6).
false.

?- is_even(6).
true.

?- is_odd(3).
true.

?- is_even(3).
false.

?-
```

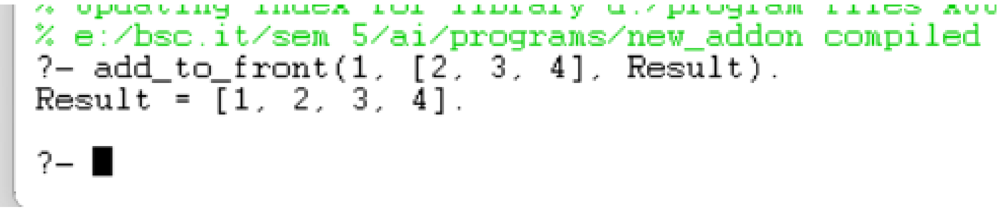
---

5) Add an element in front of the list.

Code:

```
add_to_front(Element, List, [Element|List]).
```

Output:



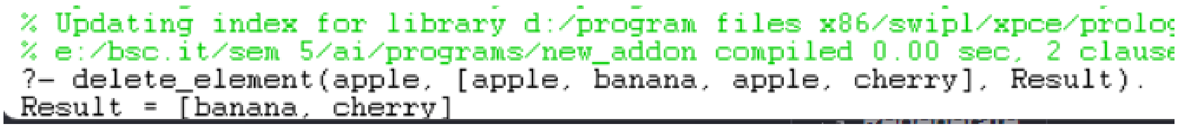
```
% updating index for library d:/program files x86
% e:/bsc.it/sem 5/ai/programs/new_addon compiled
?- add_to_front(1, [2, 3, 4], Result).
Result = [1, 2, 3, 4].

?-
```

6) Prolog code to delete an element from the list.

Code:

```
delete_element(_, [], []).

delete_element(Element, [Element|Tail], Result) :-

   delete_element(Element, Tail, Result).

delete_element(Element, [Head|Tail], [Head|Result]) :-

   Element \= Head,

   delete_element(Element, Tail, Result).
```

Output:



```
% Updating index for library d:/program files x86/swipl/xpce/prolo
% e:/bsc.it/sem 5/ai/programs/new_addon compiled 0.00 sec, 2 clause
?- delete_element(apple, [apple, banana, apple, cherry], Result).
Result = [banana, cherry]
```

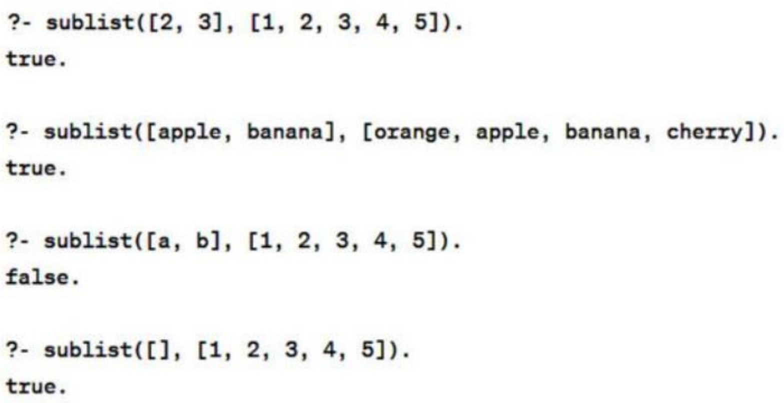
---

6) Prolog code to find a sublist from the list.

Code:

```
sublist(Sublist, List) :-

   append(_, Sublist, Rest),

   append(Sublist, _, List),

   Rest \= [].
```

Output:



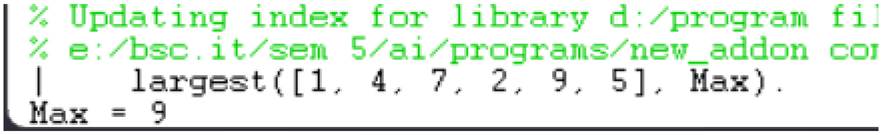
```
?- sublist([2, 3], [1, 2, 3, 4, 5]).
true.

?- sublist([apple, banana], [orange, apple, banana, cherry]).
true.

?- sublist([a, b], [1, 2, 3, 4, 5]).
false.

?- sublist([], [1, 2, 3, 4, 5]).
true.
```

6) Prolog code to find largest from the list.

Code:

```
largest([X], X).



largest([H|T], Max) :-

    largest(T, MaxSoFar),

    (H > MaxSoFar -> Max = H ; Max = MaxSoFar).
```

Output:



```
% Updating index for library d:/program fil
% e:/bsc.it/sem 5/ai/programs/new_addon co
|    largest([1, 4, 7, 2, 9, 5], Max).
Max = 9
```