AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course No: CSE 4108
Course Name: Artificial Intelligence Lab
Assignment No: 2

Submitted By:
Name:Shimul Paul
Id:160204014
Section:A
Lab Group:A1

Date of submission: 10 March,2020

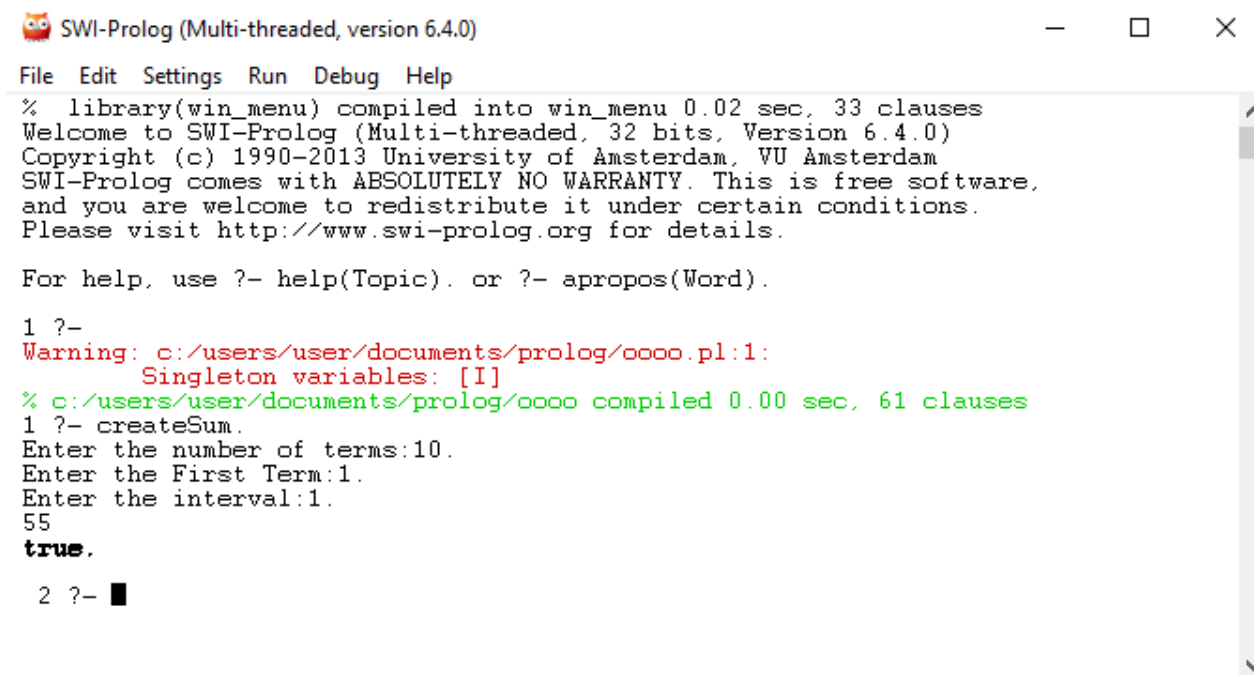
1. **Define a recursive procedure in Python and in Prolog to find the sum of 1st n terms of an equal-interval series given the 1st term and the interval.**

Prolog Code:

```
sum1(1,F,I,F):-!.
sum1(N,F,I,S):-N1 is N-1, sum1(N1,F,I,S1), S is S1+F+(N-1)*I.

createSum :- write('Enter the number of terms:'), read(N), write('Enter the First Term:'), read(F),
write('Enter the interval:'), read(I), sum1(N,F,I,S), write(S).
```

Input/Output:



Python Code:

```
def sumOfSeries(numberOfTerms, interval, firstTerm):
    if(numberOfTerms==0):
        return 0
    elif(numberOfTerms>=1):
        return sumOfSeries(numberOfTerms-1, interval,
firstTerm)+firstTerm+(numberOfTerms-1)*interval

itr = int(input('Number of iterations?'))
for i in range(itr):
    print('Iteration:', i+1)
    f = int(input('Enter First element: '))
    i = int(input('Enter Interval: '))
```

```
n = int(input('Enter number of terms: '))
print('Series sum:', sumOfSeries(n,i,f))
```

Input/Output:

```
Number of iterations?5
Iteration: 1
Enter First element: 1
Enter Interval: 2
Enter number of terms: 5
Series sum: 25
Iteration: 2
Enter First element: 1
Enter Interval: 1
Enter number of terms: 10
Series sum: 55
Iteration: 3
Enter First element: 1
Enter Interval: 2
Enter number of terms: 3
Series sum: 9
Iteration: 4
Enter First element: 1
Enter Interval: 3
Enter number of terms: 5
Series sum: 35
Iteration: 5
Enter First element: 1
Enter Interval: 5
Enter number of terms: 3
Series sum: 18
```

2. Define a recursive procedure in Python and in Prolog to find the length of a path between two vertices of a directed weighted graph.

Python Code:

```
import sys
neigh=[('i','a',35),('i','b',45),('a','c',22),('a','d',32),('b','d',28),
        ('b','e',36),('b','f',27),('c','d',31),('c','g',47),('d','g',30),('e','g',26)]
x=0
y=0
def pathlengthcheck(starting_node,ending_node):
    global w
    w=0
    i=0
    while(i<=10):
```

```
        if(neigh[i][0]==starting_node):
            if(neigh[i][1]==ending_node):
                l=neigh[i][2]
                x=i
                w=l
                return w
            else:
                print(neigh[i][1])
                return neigh[i][2]+pathlengthcheck(neigh[i][1],ending_node)

        i=i+1
starting_node=str(input('First node:'))
ending_node=str(input('destination:'))
print('path length:', pathlengthcheck(starting_node,ending_node))
```

Input/Output:

```
First node:i
destination:g
a
c
d
path length: 118
```

3. Modify the Python and Prolog codes demonstrated above to find h2 and h3 discussed above.

h2 (Manhattan distances of the tiles in the 8 puzzle problem are calculated):

Python Code:

```
initialize_source = [[1,2,3],
        [4,0,5],
        [6,7,8]]

destination_source = [[0,7,8],
        [4,5,6],
        [1,2,3]]

s = int(input())
length1=len(initialize_source)
length2=len(destination_source)
x1=0
x2=0
y1=0
```

```
y2=0

for i in range(length1):
    for j in range(length1):
        if (s == initialize_source[i][j]):
            x1 = i
            y1 = j

for i in range(length2):
    for j in range(length2):
        if (s == destination_source[i][j]):
            x2 = i
            y2 = j
v=abs(x1-x2)+abs(y1-y2)

print("Manhaten distance of ",s," : ",v)
```

Input/Output:

```
3
Manhaten distance of  3  :  2
```