Ahsanullah University of Science and Technology Department of Computer Science and Engineering

Spring 2020

Course No: CSE4108
Course Title: Artificial Intelligence Lab
Assignment No: 02

Submitted by
Name: Rizeya Rabbi Reyad
ID: 160104082
Lab Group: B2

Question 3: 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.

Series: 100+105+110+ … +(100+(n-1)x5)

Prolog Code: sum1(1, 100):-!. sum1(N, S):- N1 is N-1, sum1(N1, S1), S is S1+100+(N-1)*5.

go:- write('No. of term: '), read(N), sum1(N, S), write(S).

```
% c:/users/asus/documents/prolog/sum compiled 0.00 sec, 4 clauses
1 ?- go.
No. of term: 3.
315
true.
```

Output:

Python Code:

```
def ssum(N,I,F):
if (N==0):
return 0 elif
(N>=1):
    return ssum(N-1,I,F)+F+(N-1)*I

# Main
f=int(input('First element:'))
d=int(input('Interval:'))
n=int(input('No of term:'))
print('Series sum:', ssum(n,d,f))
```

Question 4: Define a recursive procedure in Python and in Prolog to find the length of a path between two vertices of a directed weighted graph.

Prolog Code:

```
neighbor(i,a,35).   neighbor(i,b,45).   neighbor(a,c,22).   neighbor(a,d,32).
neighbor(b,d,28).   neighbor(b,e,36).   neighbor(b,f,27).   neighbor(c,d,31).
```

```
neighbor(c,g,47).       neighbor(d,g,30).       neighbor(e,g,26).
```

```
pathLength(X,Y,L):- neighbor(X,Y,L),!. pathLength(X,Y,L):-
neighbor(X,Z,L1), pathLength(Z,Y,L2), L is L1+L2.
```

```
go(X,Y,L):- pathLength(X,Y,L), write(L).
```

Output:

```
% c:/users/asus/desktop/4pl compiled 0.00 sec, 15 clauses
1 ?- go(i,g, S).
104
S = 104 ,
```

PythonCode:

```
tupleList1 = [('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)]

def findLength(s, g):    if s
== g:       return 0    else:
for i in range(0, 10, 1):
        if tupleList1[i][0] == s and tupleList1[i][1] == g:
            return tupleList1[i][2]

    for i in range(0, 10, 1):
        for j in range(i + 1, 10, 1):
            if (tupleList1[i][0] == s and tupleList1[i][1] == tupleList1[j][0]):
                return tupleList1[i][2] + findLength(tupleList1[j][1], g)

# Main s = str(input('Starting Node:
')) g = str(input('Goal Node: '))
print('Path Length:', findLength(s, g))
```

Question 5: Modify the Python and Prolog codes demonstrated above to find h2 and h3 discussed above.

H2 Prolog Code:

```
gtp(1,1,1).      gtp(2,1,2).      gtp(3,1,3).      gtp(4,2,3).      gtp(5,3,3).      gtp(6,3,2).
 gtp(7,3,1).  gtp(8,2,1).  gblnk(2,2).  tp(1,1,2).  tp(2,1,3).  tp(3,2,1).  tp(4,2,3).  tp(5,3,3).
tp(6,2,2). tp(7,3,2). tp(8,1,1). blnk(3,1).
```

```
go:- calcH(1,[],L), sumList(L,V),write('Heuristics: '),write(V).
```

```
calcH(9,X,X):-!. calcH(T,X,Y):- dist(T,D), append(X,[D],X1), T1 is T+1,
calcH(T1,X1,Y).

dist(T,V):-tp(T,A,B), gtp(T,C,D), V is abs(A-C) + abs(B-D).

sumList([],0):-!. sumList(L,V):-L=[H|T],
sumList(T,V1), V is V1+H.
```

Output:

```
% c:/users/asus/desktop/h2 compiled 0.00 sec, 25 clauses
1 ?- go.
Heuristics: 8
true.
```

H2 Python Code:

```
gtp=[(1,1,1), (2,1,2), (3,1,3), (4,2,3), (5,3,3), (6,3,2), (7,3,1), (8,2,1)] tp=[(1,1,2),
(2,1,3), (3,2,1), (4,2,3), (5,3,3), (6,2,2), (7,3,2), (8,1,1)]

res = 0 for i in
range(0, 8, 1):
  res += abs(gtp[i][1] - tp[i][1]) + abs(gtp[i][2] - tp[i][2]) print(res)
```

H3 Prolog Code:

```
:-dynamic(hval/1).
/* Evaluates a 8-queens&#39; state given as list of 8 digits */

evalState(L,V):- assert(hval(0)),hl(1,L), di_up(1,L),di_dn(1,L),hval(V),
retractall(hval(_)). hl(8,_):-!.  hl(I,L):- nthel(I,L,X), chk_incr(I,L,X), I1
is I+1, hl(I1,L).

chk_incr(8,_,_):-!.  chk_incr(I,L,X):- I1 is I+1, nthel(I1,L,Y),
do_incr(X,Y),chk_incr(I1,L,X).

do_incr(X,Y):- X=Y, incr_hval.
do_incr(_,_).

incr_hval:-hval(V), V1 is V+1, retract(hval(_)), assert(hval(V1)).
```

```
di_up(8,_):-!.  di_up(I,L):- nthel(I,L,X), chkup_incr(I,L,X,0), I1 is
I+1, di_up(I1,L).

chkup_incr(8,_,_,_):-!. chkup_incr(I,L,X,K):- I1 is I+1, nthel(I1,L,Y), K1 is K+1, doup_incr(X,Y,K1),
chkup_incr(I1,L,X,K1).

doup_incr(X,Y,K1):- X1 is X+K1, Y=X1, incr_hval.  doup_incr(_,_,_).

di_dn(8,_):-!.
di_dn(I,L):- nthel(I,L,X), chkdn_incr(I,L,X,0), I1 is I+1, di_dn(I1,L).

chkdn_incr(8,_,_,_):-!.
chkdn_incr(I,L,X,K):- I1 is I+1, nthel(I1,L,Y), K1 is K+1, dodn_incr(X,Y,K1), chkdn_incr(I1,L,X,K1).

dodn_incr(X,Y,K1):- X1 is X-K1, Y=X1, incr_hval.
dodn_incr(_,_,_).

% A procedure to find the nth element of a list
nthel(N,[_|T],El):-  N1  is  N-1,  nthel(N1,T,El).
nthel(1,[H|_],H):-!.
```

H3 Python Code:

```
count = 0
arr = [[0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 1, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 1]]

def findPair(row, col):
global count    #same row
attack    for i in range (col+1,
8, 1):       if i<8 and
arr[row][i]==1:
        count = count + 1
#diagonal up attack    tcol =
col+1    for i in range(row-1,
-1, -1):       if tcol>7:
break       if arr[i][tcol] == 1:
count = count + 1       tcol =
tcol + 1
  #diagonal down attack
tcol = col+1    for i in
```

```
range(row+1, 8, 1):       if
tcol>7:           break       if
arr[i][tcol] == 1:
count = count + 1       tcol
= tcol + 1

#Main
for j in range(0, 8, 1):
for i in range(0, 8, 1):
if arr[i][j] == 1:
findPair(i, j)

print(count)
```