Artificial Intelligence

Nehal Jhajharia (U20CS093)
Lab Assignment 6

Q1.Write a PROLOG program based on list:-
A) To find the length of a list.
B) To find whether a given element is a member of a list.
C) To add the member of a given list (sum of elements of List).
D) To find the last element of a list.
E) To reverse a list.
Code:

```
% list length
lisLen([], 0).
lisLen([_|TAIL], LEN) :- lisLen(TAIL, N), LEN is N+1.
% membership test
isLisMem(X, [X|_], Found):- Found is 1.
isLisMem(X, [_|T], Found) :- isLisMem(X, T, Found).
% add members
addLisMem([], 0).
addLisMem([H|T], SUM) :- addLisMem(T, ST), SUM is H + ST.
% last element
lisLasEle([H], LE) :- LE is H.
lisLasEle([_|T], LE) :- lisLasEle(T, LE).
% Reverse a list
lisConcat([],L,L).
lisConcat([H1|T1],L2,[H1|T3]) :- lisConcat(T1,L2,T3).
lisRev([],[]).
lisRev([H|T],RL) :- lisRev(T, RT),lisConcat(RT, [H],RL).
main:-
    write("The list: [1, 2, 3]"),nl,
    write("Length of list is "),
    lisLen([1, 2, 3], LEN),
            write(LEN),nl,
            write("Is 1 member of list: "),
            isLisMem(1, [1, 2, 3], Found),
    write(Found), nl,
    write("Sum of list: "),
    addLisMem([1, 2, 3], SUM),
    write(SUM), nl,
    write("Last element of list: "),
    lisLasEle([1, 2, 3], LE),
    write(LE), nl,
    write("Reversed list: "),
    lisRev([1, 2, 3], RL),
    write(RL),nl.
```

The list: [1, 2, 3]

Length of list is 3

Is 1 member of list: 1

Sum of list: 6

Last element of list: 3

Reversed list: [3, 2, 1]

 true

| Next | 10 | 100 | 1,000 | Stop |

?-   main

Q2. Implement A* algorithm. You can implement the algorithm in any language. Try to solve following problem as shown figure 1 below -Find shortest path from A to J

```python
import heapq

def astar(graph,start_node,end_node):
    f_distance={node:float('inf') for node in graph}
    f_distance[start_node]=0
    g_distance={node:float('inf') for node in graph}
    g_distance[start_node]=0
    came_from={node:None for node in graph}
    came_from[start_node]=start_node
    queue=[(0,start_node)]
    while queue:

        current_f_distance, current_node = heapq.heappop(queue)

        if current_node == end_node:
```

```python
        return f_distance, came_from

    for next_node,weights in graph[current_node].items():
        temp_g_distance = g_distance[current_node] + weights[0]

        if temp_g_distance<g_distance[next_node]:
            g_distance[next_node]=temp_g_distance
            heuristic=weights[1]
            f_distance[next_node]=temp_g_distance+heuristic
            came_from[next_node]=current_node

            heapq.heappush(queue, (f_distance[next_node], next_node))

return f_distance, came_from


if __name__ == "__main__":
    # node: [weight, h(n)]
    graph = {
        'A':{'B':[6,8],'F':[3,6]},
        'B':{'C':[3,5],'D':[2,7]},
        'C':{'D':[1,7], 'E':[5,3]},
        'D':{'E': [8,3]},
        'E':{'I':[5,1], 'J':[5,0]},
        'F':{'G':[1,5], 'H':[7,3]},
        'G':{'I':[3,1]},
        'H':{'I':[2,1]},
        'I':{'J': [3, 0]},
        'J':{}
    }

    start_node = 'A'
    end_node = 'J'

    ret = astar(graph, start_node, end_node)

    path = [end_node]
    cur_node = end_node
    while cur_node != start_node:
        cur_node = ret[1][cur_node]
        path.append(cur_node)
```

```python
    path.reverse()

print("Path: ", path)
```

jhajharia@Nehals-MacBook-Air Asmt5 % python3 main.py
Path: ['A', 'F', 'G', 'I', 'J']
jhajharia@Nehals-MacBook-Air Asmt5 %