# **Team Members**

## Name: Mahaboob Ali Ashraf Mohammad, NetID: mmoha055

## Name: Anvaya, NetID: an001

Week 5 (Due Thursday, November 02 by midnight)

**Question 1 (3 points): Finding a Fixed Food Dot using Depth First Search**Through the task of implementing depth-first search, I learned how to apply and optimize this fundamental search algorithm in a real-world context. It provided a hands-on understanding of the intricacies of managing a search stack, efficiently backtracking when necessary, and the trade-offs between recursion and iteration. I also grasped the importance of graph search to prevent revisiting states and gained insights into code optimization to prevent issues like stack overflow. Overall, this experience deepened my knowledge of algorithms, honed my problem-solving and coding skills, and underscored the significance of testing and documentation in software development.  
Credit: Used ChatGPT to get the alternative idea of using the while loop as I was hitting the recursion stack Limit. **- Solution for the 3 example mazes  
A screenshot of a computer screen

Description automatically generated  
- Auto grader results  
A screenshot of a computer program

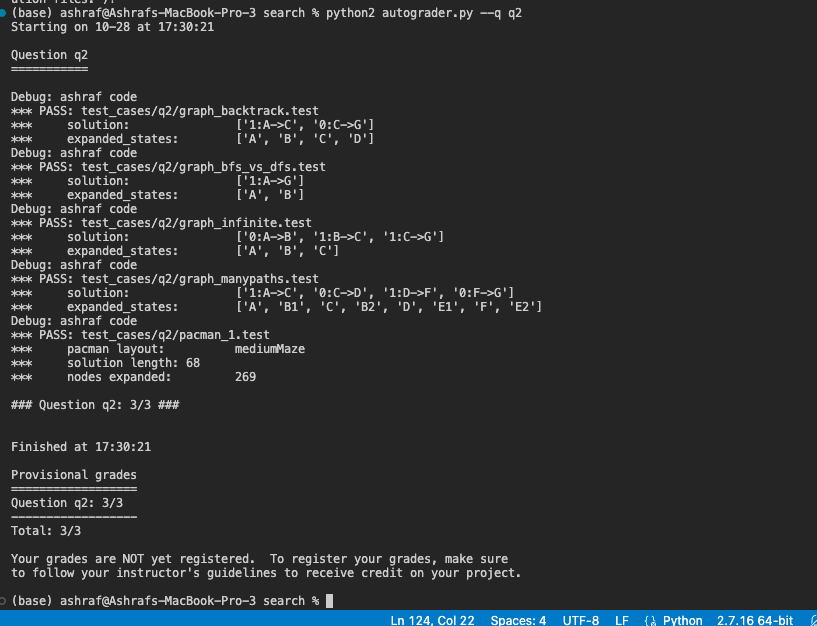
Description automatically generated**

**Question 2 (3 points): Breadth First Search**

Through implementing Breadth-First Search (BFS) for Question 2, I learned the fundamental difference between BFS and Depth-First Search (DFS). BFS prioritizes exploring all nodes at the current level before moving deeper, which ensures it finds the shortest path to the goal in unweighted graphs. The use of a queue for exploration, pushing nodes to the back, and popping from the front allows BFS to guarantee optimality. This experience emphasized the importance of choosing the right search algorithm based on the problem's characteristics, like weighted or unweighted graphs and the need for the shortest path. It also highlighted the similarities in code structure between BFS and DFS, where the key distinction lies in the order of node exploration and the data structure used.

**- Solution for the 3 example mazes**

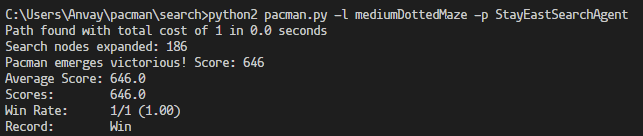
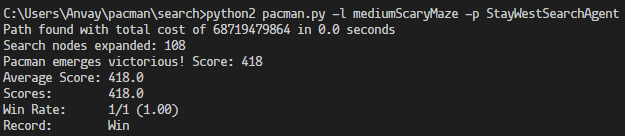
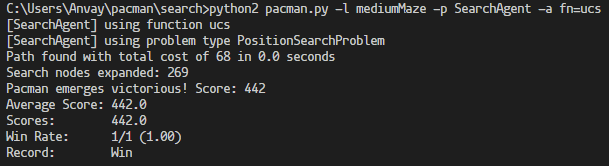
**A screenshot of a computer

Description automatically generated  
  
- Auto Grader Result**

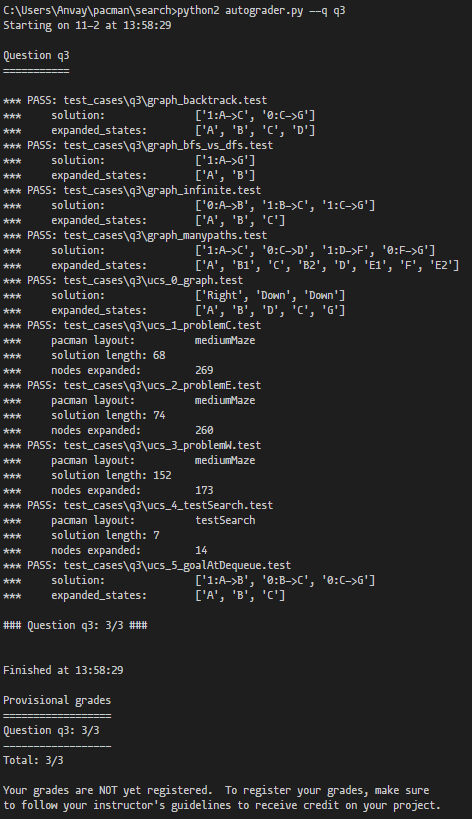
**Question 3 (3 points): Uniform Cost Search**

Uniform cost search or UCS is a cost-based search algorithm which ensures the lowest path retrieval in a search space. The efficiency of UCS is influenced by how accurately costs are assigned to each action/step. The performance of UCS can also be enhanced with the use of data structures such as priority queue and sets. To store and retrieve the path, cost and to keep track of the visited states of the Pacman.

The Key Observation here was from the snippets below was **StayEastSearchAgent** and **StayWestSearchAgent** have a very low and very high-cost function due to there exponential cost function.

****

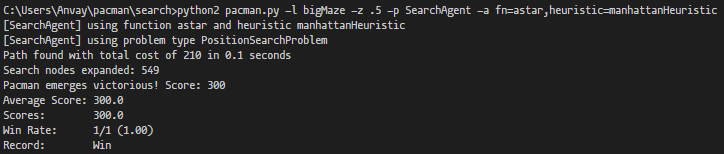
**Auto-Grader Results**

****

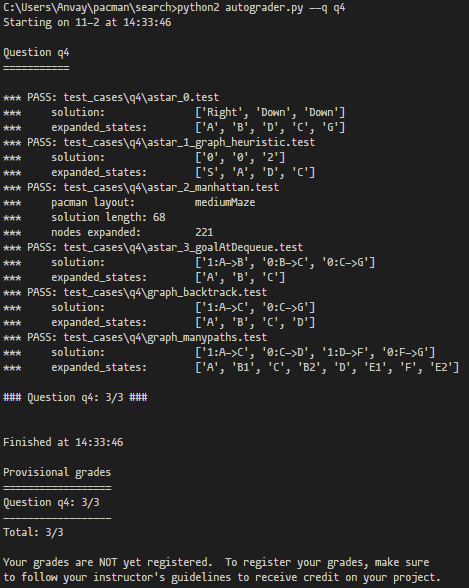
**Question 4 (3 points): A\* search**

The A\* search that was implemented is very similar to the Uniform Cost Search Function with the use of Priority queue. However, here the priority to different action/step is allotted differently. With, the use of heuristic functions.

The total cost path is 210 with search space of 549 nodes.

****

**Auto-Grader Results**

****