LAB 2

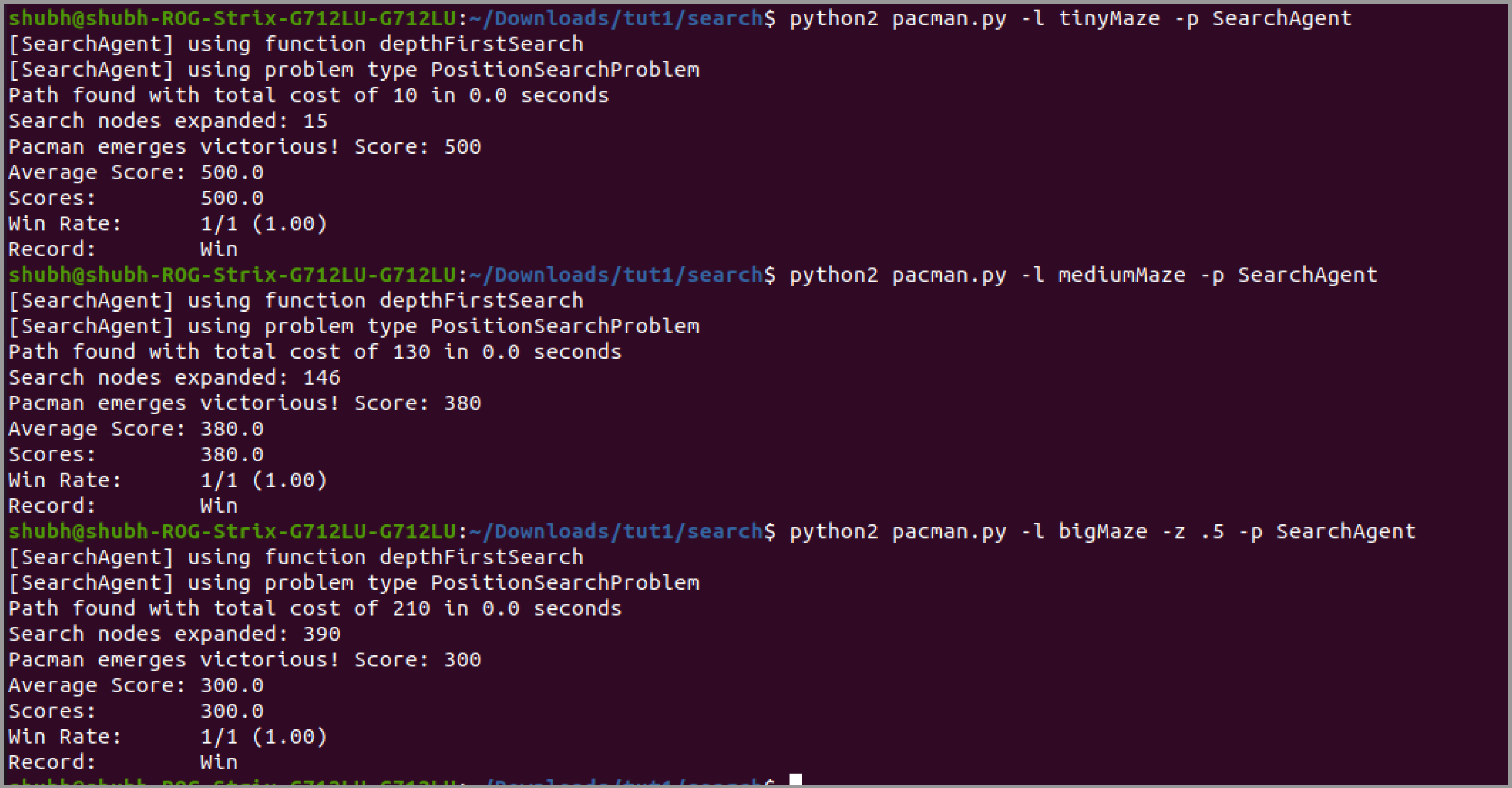
BY- SHUBH PAREEK
112001039

This file contains justifications,results and explanations
For the code written in questions by me

Q1

Normal dfs is implemented in it, stack is taken from the utility.py given in Files.
For tinymaze my dfs expands 15 nodes, and scores 500
For mediummaze it expands 130 nodes, and scores 300
For bigmaze it expands 390 nodes, and scores 390

Required proof



```
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l tinyMaze -p SearchAgent
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 10 in 0.0 seconds
Search nodes expanded: 15
Pacman emerges victorious! Score: 500
Average Score: 500.0
Scores:        500.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l mediumMaze -p SearchAgent
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 130 in 0.0 seconds
Search nodes expanded: 146
Pacman emerges victorious! Score: 380
Average Score: 380.0
Scores:        380.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l bigMaze -z .5 -p SearchAgent
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 390
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$
```

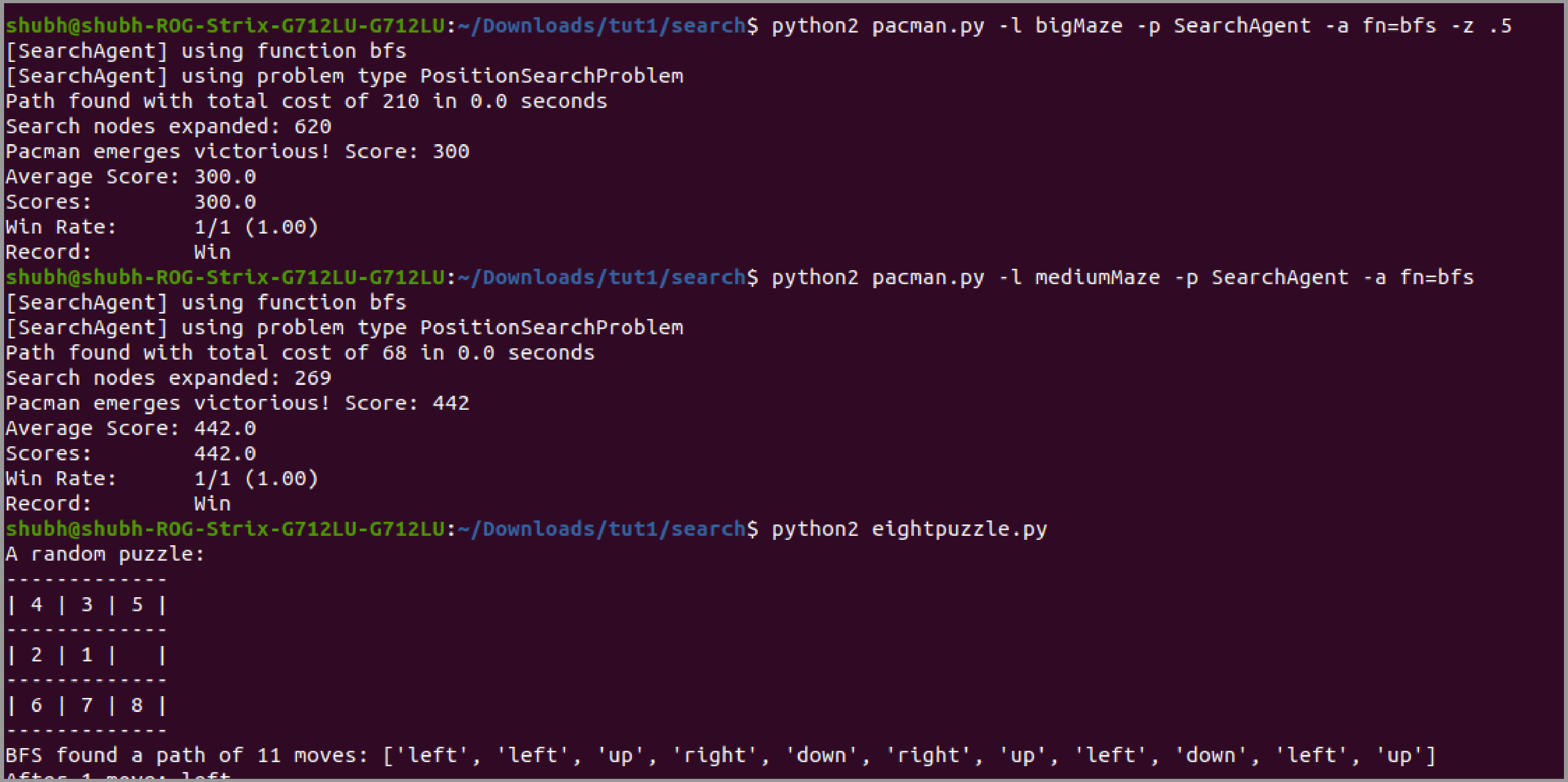
Q2

Normal bfs is implemented with a queue function imported from util.py file. For bigmaze my implementation expands 620 nodes ,with score of 300

For mediummaze my implementation expands 269 nodes ,with score of 442

It solves eightpuzzel in 11 moves.

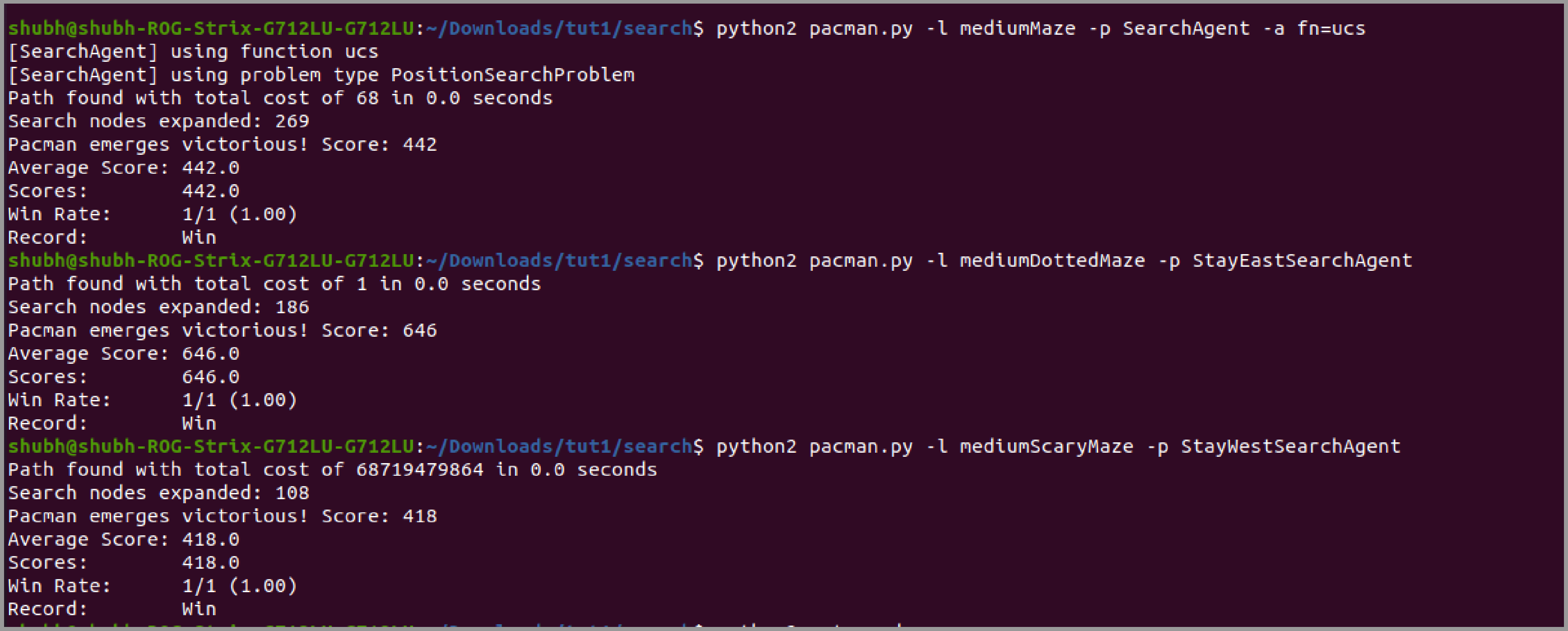
Required proofs



```
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 620
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l mediumMaze -p SearchAgent -a fn=bfs
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 eightpuzzle.py
A random puzzle:
-------------
| 4 | 3 | 5 |
-------------
| 2 | 1 |   |
-------------
| 6 | 7 | 8 |
-------------
BFS found a path of 11 moves: ['left', 'left', 'up', 'right', 'down', 'right', 'up', 'left', 'down', 'left', 'up']
```

Q3

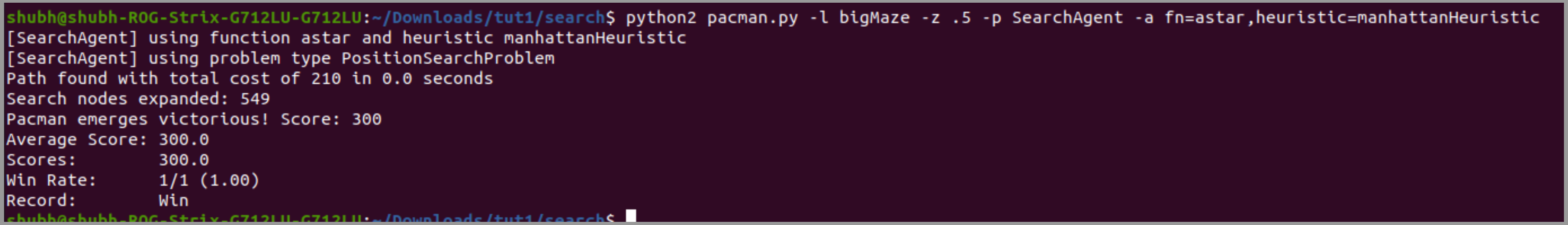
Uniform cost search implementation results are



```
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l mediumMaze -p SearchAgent -a fn=ucs
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l mediumDottedMaze -p StayEastSearchAgent
Path found with total cost of 1 in 0.0 seconds
Search nodes expanded: 186
Pacman emerges victorious! Score: 646
Average Score: 646.0
Scores:        646.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l mediumScaryMaze -p StayWestSearchAgent
Path found with total cost of 68719479864 in 0.0 seconds
Search nodes expanded: 108
Pacman emerges victorious! Score: 418
Average Score: 418.0
Scores:        418.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Q4

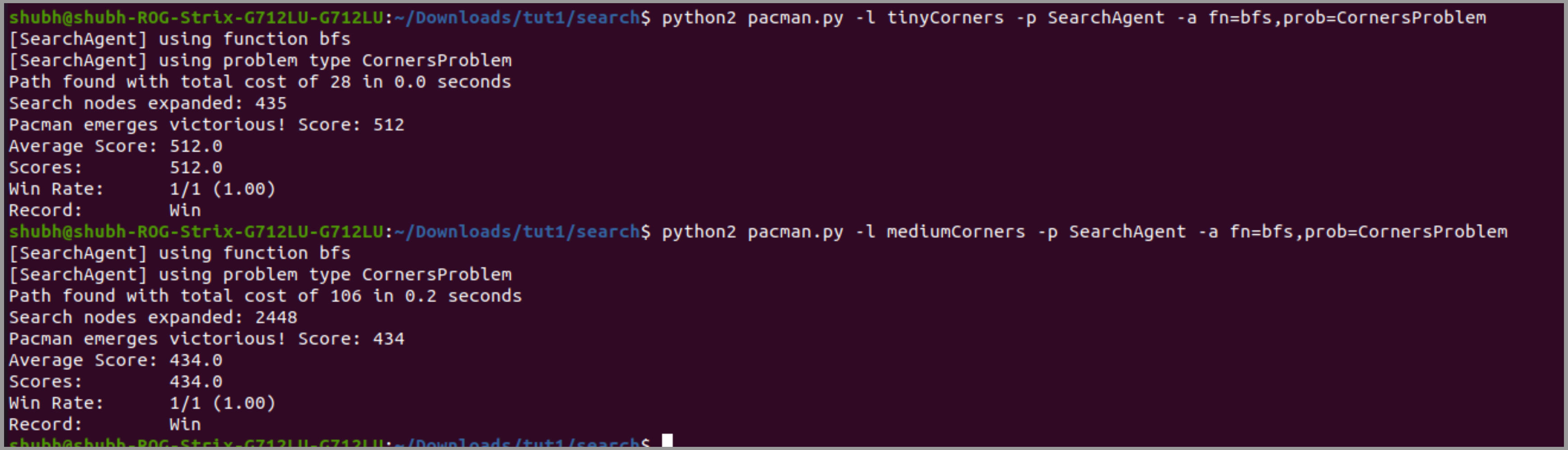
A* search results with manhattan heuristic are



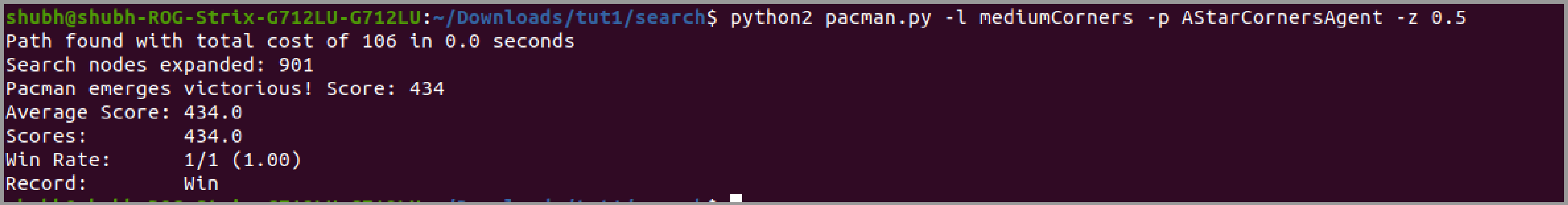
```
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 549
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$
```

Q5 + Q6

Corners problem implementation results are



```
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
[SearchAgent] using function bfs
[SearchAgent] using problem type CornersProblem
Path found with total cost of 28 in 0.0 seconds
Search nodes expanded: 435
Pacman emerges victorious! Score: 512
Average Score: 512.0
Scores:        512.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
[SearchAgent] using function bfs
[SearchAgent] using problem type CornersProblem
Path found with total cost of 106 in 0.2 seconds
Search nodes expanded: 2448
Pacman emerges victorious! Score: 434
Average Score: 434.0
Scores:        434.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$
```



```
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5
Path found with total cost of 106 in 0.0 seconds
Search nodes expanded: 901
Pacman emerges victorious! Score: 434
Average Score: 434.0
Scores:        434.0
Win Rate:      1/1 (1.00)
Record:        Win
```

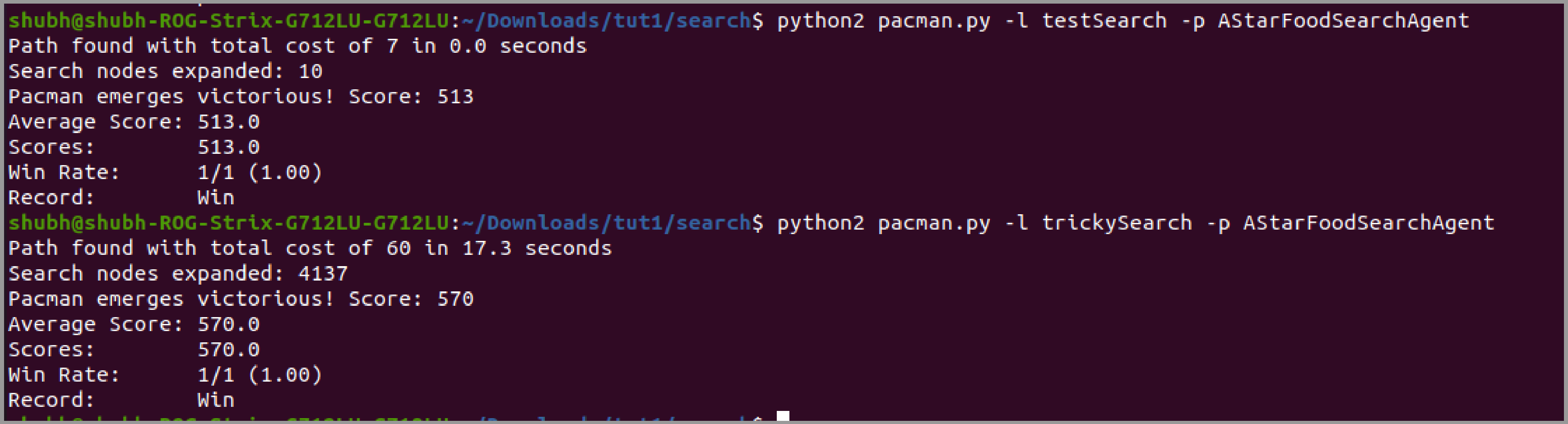
Heuristic used -

The heuristic implemented calculates the nearest corner(in terms of manhattan distance) from the current point and similarly the next location is determined on the basis of manhattan distance ,so total cost returned is the sum of manhattan distance between two successive(in terms of shortest manhattan distance) corners in the path.

Now this heuristic is admissable because it will be always less than or equal to the actual cost of completing the whole path, the correctness is justified by the fact that manhattan distance is the minimum distance between two points on the grid if you move only horizontally and vertically. It is also consistent because the heuristic cost of the proceeding path never exceeds prediction by the parent heuristic cost .

Q7.

Results of eating all foods problem -



```
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l testSearch -p AStarFoodSearchAgent
Path found with total cost of 7 in 0.0 seconds
Search nodes expanded: 10
Pacman emerges victorious! Score: 513
Average Score: 513.0
Scores:        513.0
Win Rate:      1/1 (1.00)
Record:        Win
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Downloads/tut1/search$ python2 pacman.py -l trickySearch -p AStarFoodSearchAgent
Path found with total cost of 60 in 17.3 seconds
Search nodes expanded: 4137
Pacman emerges victorious! Score: 570
Average Score: 570.0
Scores:        570.0
Win Rate:      1/1 (1.00)
Record:        Win
```

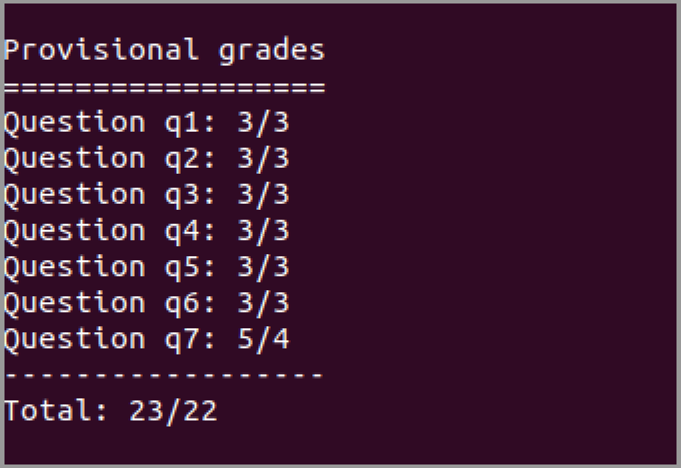
Heuristic used -

The way I calculate heuristic cost is by selecting the maximum cost of minimum cost path (bfs) to each food. Because even in the best case(when all fruits are in a single path that is from starting point till last food in path) we will still need to cover the farthest food. So this ensures that the heuristic cost will always be less than the actual cost as it imitates the best case cost, hence admissible.

I used mazeDistance which is an already implemented function that calculates the shortest path between two points in a given state.

This heuristic gets a grade of 5/4 in the auto grader , whereas when I used Manhattan distance it got 0/4 in the auto grader .

<u>RESULT OF AUTOGRADER</u>



```
Provisional grades
==================
Question q1: 3/3
Question q2: 3/3
Question q3: 3/3
Question q4: 3/3
Question q5: 3/3
Question q6: 3/3
Question q7: 5/4
------------------
Total: 23/22
```

<u>Thank you</u>