COMP105 – Assignment 3 Feedback

Username: sgtentic
Overall Mark: 74/100

Below you will find a test report for each function that you submitted. Each test is broken down into a number of test sets, that each try to test different aspects of your function.

Test inputs. The tests use maze2 through maze6, and maze-big-1 through maze-big-4, which are the same mazes that were distributed with the handout. In addition to these, the following mazes are also used: maze7 is a previously unseen small maze:

```
###############
#             #
############# #
#     #       #
### # ### ### #
#   #   # #   #
### # # # # ###
#   # #   #   #
########## ###
# #          #
# ########## #
#   #        #
# # # ##### ###
# #   #       #
###############
```

maze-big-5 is a previously unseen big maze:

```
#################################################################
#     #   #       # #   # #         #           #       # # #
### # # # ##### ### ### # # # ### # # ########## # ##### # #
#   # # #         #     #   #   # #          #   #     #   #
### # # # # ####### # ### ### ##### ##### ##### # # ##### # #
#   # # # #       # #   #   #   # # #   #       # #     # # #
# # # # ### # ### # ### ### # # # # # # ### # # # # ### # ###
# # #     # #   #   #       # #   # # #     # # # #   #   # #
### ########################################################## # #
#     # #   #     #   # #         #       # # # # # # # # # #
# # ### # # ##### # # # ####### ### ### # # # # # # # # # # #
# #     # #   # # # #             #   # # # #   # # # # #   #
# # ### ### # # # # ### ##### ####### ### # # ### # # # ### #
# #   #     #         # #         #   #   # #     #     #   #
# # ### # # # ######### ##### ####### # # # # ### # # # ### #
# #   # # # #         # #         #   # #       # # # # # # #
##### ### ############ ################ ### ### ##### ### # # #
# #   # #   #               #         #   # #         # # # #
# ### # # # # # ### ##### # # # # ### ### # # # ### ### # # #
```

```
#     #   # # # #   #     #   # # #   #   #   #   #   # #   #
#################################################################
```

and maze1 is the following maze

```
###
# #
###
```

This maze is a challenging edge case. The handout promises that the maze will be entirely surrounded by walls, so an empty maze is not a valid input, and therefore maze1 is the smallest valid maze.

Question 1: get_maze

```
Test: 1
=======
Called: get_maze "maze1.txt"
Output:
["###","# #","###"]
Result: Correct


Test: 2
=======
Called: get_maze "maze2.txt"
Output:
["#####","#   #","# # #","# # #","#####"]
Result: Correct


Test: 3
=======
Called: get_maze "maze7.txt"
Output:
["################","#              #","############## #","#     #       #","### # ...
Result: Correct


Test: 4
=======
Called: get_maze "maze-big-5.txt"
Output:
["#####################################################################","#     #   #   ...
Result: Correct
```

Mark: 10/10

The most common cause of problems for get_maze is hard-coding a path into the function. The auto-marker runs the function with the mazes in the current working directory, so if get_maze "maze2.txt" is called, then the maze should be loaded in. If you have hard coded a directory into the function, then you will fail these tests.

Question 2: print_maze

```
Test: 1
=======
Called: print_maze maze1.txt
Output:
```

```
###
# #
###
```

```
Result: Correct
```

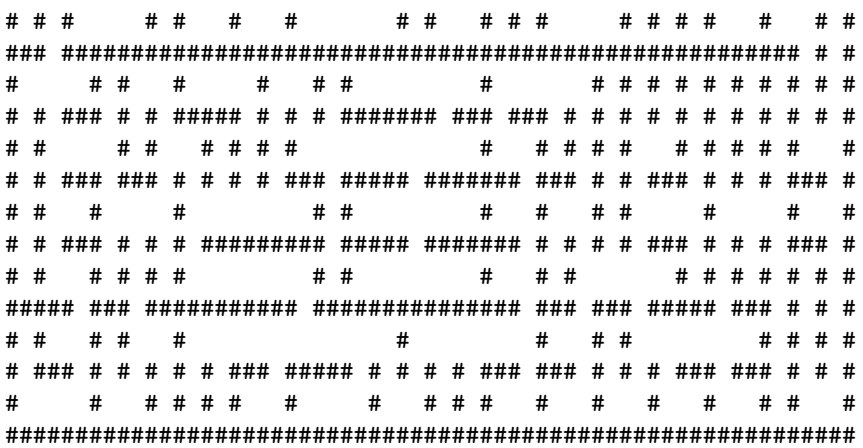
```
Test: 2
=======
Called: print_maze maze2.txt
Output:
#####
#   #
# # #
# # #
#####
```

```
Result: Correct
```

```
Test: 3
=======
Called: print_maze maze7.txt
Output:
################
#              #
############## #
#     #        #
### # ### ### #
#   #   # #   #
### # # # # ###
#   # #   #   #
########### ###
# #           #
# ########### #
#   #         #
# # # ##### ###
# #   #       #
################
```

```
Result: Correct
```

```
Test: 4
=======
Called: print_maze maze-big-5.txt
Output:
#####################################################################
#     #   #       # #   # #         #           #       # # #
### # # # ##### ### ### # # # ### # # ########### # ##### # #
#   # # #         #     #   #   # #           #   #     #   #
### # # # # ####### # ### ### ##### ##### ##### # # ##### # #
#   # # # #       # #   #   #   # # #   #       # #     # # #
# # # # ### # ### # ### ### # # # # # # ### # # # # ### # ###
```



```
Result: Correct
```

Mark: 10/10

A common issue for print_maze is not printing out the final blank line. The handout specified that a blank line should be produced after the maze. If you used unlines then this would have happened automatically. If you used recursion, then you would have to make sure that your base case prints out the newline character.

Question 3: is_wall

Test Set	Argument	Output	Expected	Result	Mark
1	`maze2 (0, 1)`	`True`	`True`	`Correct`	5/5
	`maze3 (1, 1)`	`False`	`False`	`Correct`	
	`maze4 (3, 6)`	`True`	`True`	`Correct`	
	`maze-big-1 (15, 15)`	`False`	`False`	`Correct`	
	`maze-big-5 (30, 10)`	`True`	`True`	`Correct`	

Mark: 5/5

Test Set 1: This test set checks whether the function works.

Question 4: place_player

Test Set	Argument	Output	Expected	Result	Mark
1	`maze1 (1, 1)`	`["###","#@#","###"]`	`["###","#@#","###"]`	`Correct`	5/5
	`maze2 (3, 3)`	`["#####","#   #",...`	`["#####","#   #",...`	`Correct`	
	`maze4 (5, 1)`	`["#########","#  ...`	`["#########","#  ...`	`Correct`	
	`maze-big-1 (13, 5)`	`["###############...`	`["###############...`	`Correct`	
	`maze-big-5 (29, 9)`	`["################...`	`["################...`	`Correct`	

Mark: 5/5

Test Set 1: This test set checks whether the function works.

Question 5: move

Test Set	Argument	Output	Expected	Result	Mark
1	(3,1) 'w'	(3,0)	(3,0)	Correct	5/5
	(100,100) 's'	(100,101)	(100,101)	Correct	
	(123,456) 'a'	(122,456)	(122,456)	Correct	
	(999,999) 'd'	(1000,999)	(1000,999)	Correct	
2	(1,1) 'p'	(1,1)	(1,1)	Correct	4/4
	(3,3) '@'	(3,3)	(3,3)	Correct	

Mark: 9/9

Test Set 1: This test set checks whether the function works for wasd inputs.

Test Set 2: This test set checks whether the function works for non-wasd inputs. The handout specified that the point should not be moved in this case.

Question 6: can_move

Test Set	Argument	Output	Expected	Result	Mark
1	maze1 (1, 1) 'w'	False	False	Correct	9/9
	maze2 (3, 3) 's'	False	False	Correct	
	maze-big-1 (13, 5) 'd'	True	True	Correct	
	maze-big-5 (29, 9) 'a'	True	True	Correct	

Mark: 9/9

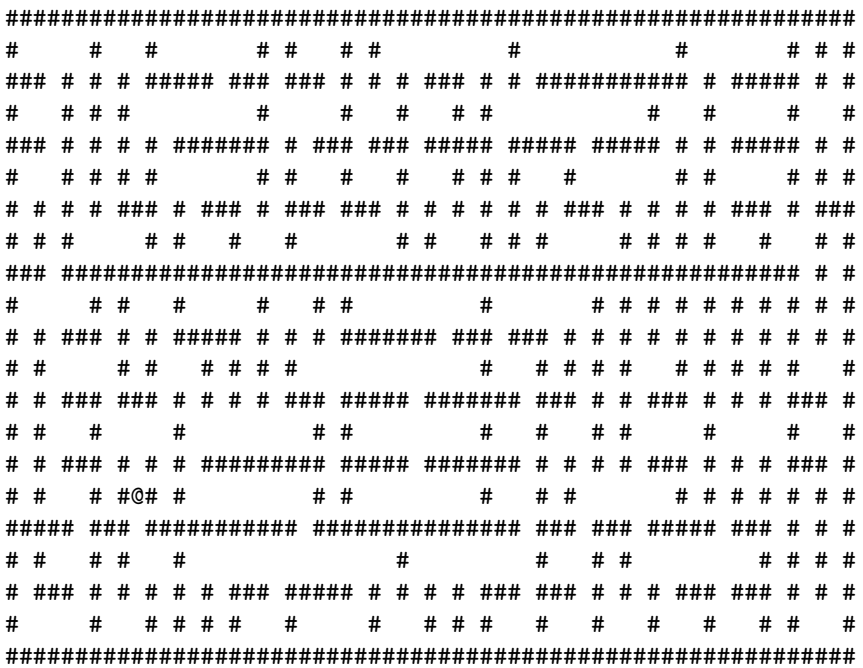
Test Set 1: This test set checks whether the function works.

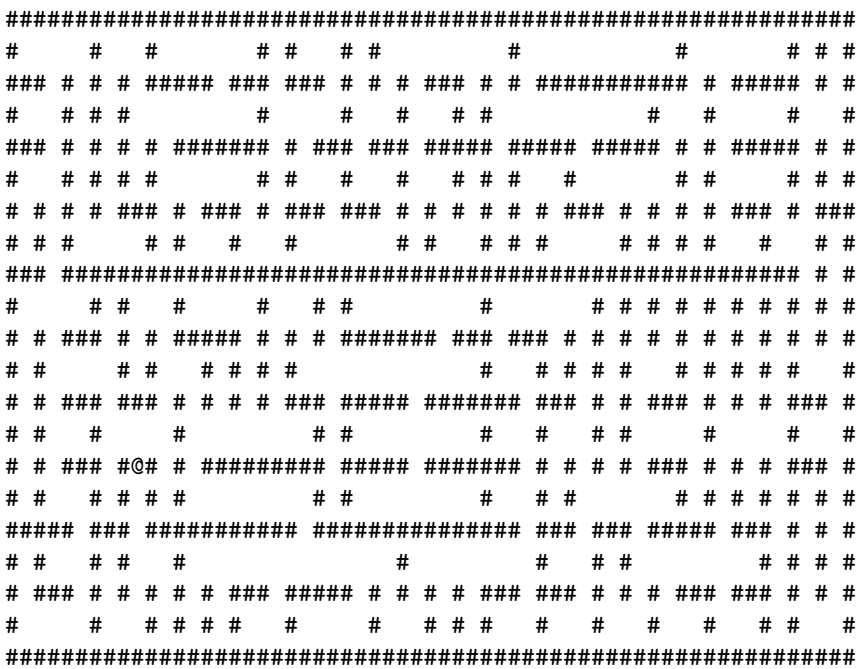
Question 7: game_loop

```
Test: 1
=======
Called: game_loop maze2 (1, 1)
Keyboard input: "s\ns\nw\n"
Output:
#####
#@  #
# # #
# # #
#####

#####
#   #
#@# #
# # #
#####

#####
#   #
# # #
#@# #
#####

#####
#   #
#@# #
# # #
```

```
#####


Result: Correct

Test: 2
=======
Called: game_loop maze3 (3, 3)
Keyboard input: "d\na\n"
Output:
#######
# #   #
# ### #
#  @  #
# ### #
# #   #
#######

#######
# #   #
# ### #
#   @ #
# ### #
# #   #
#######

#######
# #   #
# ### #
#  @  #
# ### #
# #   #
#######


Result: Correct

Test: 3
=======
Called: game_loop maze1 (1, 1)
Keyboard input: "w\na\n"
Output:
###
#@#
###

###
#@#
###

###
#@#
###


Result: Correct
```

```
Test: 4
=======
Called: game_loop maze2 (1, 1)
Keyboard input: "asdf\ndog\n"
Output:
#####
#@  #
# # #
# # #
#####

#####
#@  #
# # #
# # #
#####

#####
# @ #
# # #
# # #
#####


Result: Correct


Test: 5
=======
Called: game_loop maze4 (5, 5)
Keyboard input: "wasd\n"
Output:
#######
# #   #
# ### #
#     #
# ### #
# #  @#
#######

#######
# #   #
# ### #
#     #
# ###@#
# #   #
#######


Result: Correct


Test: 6
=======
Called: game_loop maze-big-5 (9, 15)
Keyboard input: "w\n"
Output:
```





```
Result: Correct
```

Mark: 15/15

Note that the keyboard inputs are not shown in the test results, but they are displayed at the start of each test. Tests 1 and 2 check whether the function works. Test 3 checks that nothing happens when the player attempts to walk into walls. Tests 4 and 5 check whether the function correctly ignores everything except the first character of the string. Tests 3 and 5 and 6 all check whether the function works when initialized with a point other than (1, 1).

Question 8: get_path

Test Set	Argument	Output	Expected	Result	Mark
1	`maze2 (1, 1) (3, 3)`	`[(1,1),(2,1),(3,1...`	`[(1,1),(2,1),(3,1...`	Correct	4/7
	`maze3 (1, 1) (5, 5)`	`[(1,1),(1,2),(1,3...`	`[(1,1),(1,2),(1,3...`	Correct	
	`maze4 (1, 1) (7, 7)`	`[(1,1),(1,2),(1,3...`	`[(1,1),(1,2),(1,3...`	Correct	
	`maze6 (11, 11) (1, 1)`	`Exception: Prelud...`	`[(11,11),(11,10),...`	Incorrect	
	`maze7 (13, 13) (1, 1)`	`Exception: Prelud...`	`[(13,13),(12,13),...`	Incorrect	
2	`maze-big-1 (1, 1) (59, 19)`	`Timeout`	`[(1,1),(1,2),(1,3...`	Incorrect	0/5
	`maze-big-2 (1, 1) (59, 19)`	`Timeout`	`[(1,1),(2,1),(3,1...`	Incorrect	
	`maze-big-3 (1, 1) (59, 19)`	`Timeout`	`[(1,1),(2,1),(3,1...`	Incorrect	
	`maze-big-4 (59, 19) (1, 1)`	`Exception: Prelud...`	`[(59,19),(58,19),...`	Incorrect	
	`maze-big-5 (59, 19) (1, 1)`	`Exception: Prelud...`	`[(59,19),(59,18),...`	Incorrect	
3	`maze1 (1, 1) (1, 1)`	`[(1,1)]`	`[(1,1)]`	Correct	5/5
	`maze-big-3 (3, 3) (3, 3)`	`[(3,3)]`	`[(3,3)]`	Correct	
	`maze-big-5 (59, 19) (59, 19)`	`[(59,19)]`	`[(59,19)]`	Correct	

All tests passed bonus: 0/5
Mark: 9/22

Test Set 1: This test set checks whether the function works for small mazes.

Test Set 2: This test set checks whether the function works for large mazes.

Test Set 3: This test set checks whether the function works when it is asked to find a path from a point to itself. Here, since the handout asks for the path to include the start and end points, the correct answer is a list containing the point itself and nothing else.

Question 9: main

```
Test: 1
=======
Called: main "maze7.txt"
Output:
###############
#.............#
#############.#
#     #      .#
### # ### ###.#
#   #   # #...#
### # # # #.###
#   # #   #.  #
###########.###
# #        ...#
# ###########.#
#   #      ...#
# # # #####.###
# #   #    ...#
###############


Result: Correct


Test: 2
=======
```

```
Called: main "maze-big-5.txt"
Output:
Timeout
Result: Incorrect


Test: 3
=======
Called: main "maze1.txt"
Output:
###
#.#
###


Result: Correct
```

Mark: 7/10

Test 1 checks a small maze, while test 2 checks a big maze. Test 3 checks maze1. In this maze, the top left and bottom right points are the same point: (1, 1), So the correct output needs to put a dot on the point (1, 1), which is both the start and end of the path.

The submission was 1 day late, for which the late penalty is -**5 marks**.

Perfect Part C bonus: 0/5

Overall Mark 74/100