Lab 6

DSA BS DS Fall 2022

Task 1

Write a function that solves a maze problem. In this particular version of the problem, a hiker must find a path to the top of a mountain. Assume that the hiker leaves a parking lot, marked **P**, and explores the maze until he reaches the top of a mountain, marked **T**.

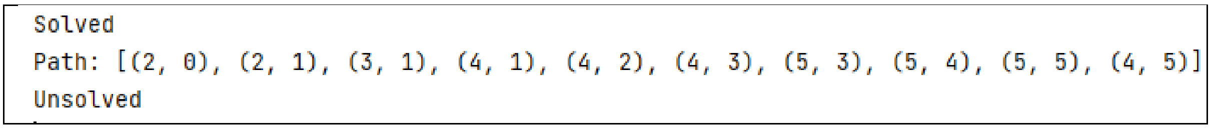
```
**************************************************
*******                              ******** ****
******* ************** ************* ******** ****
******* ************** ***         * ***      ****
P       ************** **   ****** * *** **** ****
******* ***         ** *   *******       **** ****
******* *** ******* ** *  ******************* ****
******* *** ******* **    ******************* ****
******* *** ******* ************************* ****
******* *** **                      ********* ****
*** ***     ** **** **** ******************** ****
*** ********** **** ****               ****** ****
*** ********** **** ************************* ****
*** ********** **** ************************* ****
***            **** ************ ************ ****
******** ********** ************ ************ ****
******** ********** ************      ******* ****
********      ***** ************ **** ******* ****
*******************              **** ******* ****
************************************* ******* ****
************************************* ************
*************************************            T
**************************************************
```

The maze is represented by a 2D array of characters, the character * marks a barrier, and P and T mark the parking lot and mountain top, respectively. A blank space marks a step along a path. Your program then attempts to find a path through the maze and returns "solved" or "unsolved" depending on the outcome and if the maze is solved it should print the complete path.

You have to design a **solve_maze**(maze) function. Test your function using the following driver program

```
def solve_maze(maze):
    # Write your code here.


# Driver
maze = [
    [" ", "*", " ", "*", " ", " "],
    [" ", "*", " ", "*", " ", " "],
    ["P", " ", " ", " ", "*", " "],
    ["*", " ", "*", "*", "*", " "],
    [" ", " ", " ", " ", "*", "T"],
    ["*", " ", " ", " ", " ", " "]
]

status, path = solve_maze(maze)
print(status)
if status == "Solved":
    print("Path:", path)


maze = [
    [" ", "*", " ", "*", " ", " "],
    [" ", "*", " ", "*", " ", " "],
    ["P", " ", " ", " ", "*", " "],
    ["*", " ", "*", "*", "*", " "],
    [" ", " ", " ", " ", "*", "T"],
    ["*", " ", " ", " ", " ", "*"]
]

status, path = solve_maze(maze)

print(status)
if status == "Solved":
    print("Path:", path)
```

The output of the following program should be



```
Solved
Path: [(2, 0), (2, 1), (3, 1), (4, 1), (4, 2), (4, 3), (5, 3), (5, 4), (5, 5), (4, 5)]
Unsolved
```