### Name: Tradd Schmidt

### As a lab, we require you to answer the following prompts. Place your responses in the boxes below.

### INITIAL DESIGN PLAN: What is a pseudocode design plan which meets the computational requirements of this lab? In other words, describe what the problem is, and include an outline of your solution (NOT CODE!) with enough details so that someone else could implement it for you in any language. Therefore, you must make sure that this design is not specific to Python or C++ and should use only ideas such as conditional statements, loops, and lab statements.

### You should do this before you implement your program.

|  |
| --- |
| The text of the file will need to be used as the map. The position of the M in the text is your starting position. The M will keep changing until all paths have been explored, or the T representing the treasure is found. When the M moves, it will need to determine where to move. If there is only one “.” in all of the four directions, then that is the only place it can go. Otherwise, some sort of marker needs to be used to indicate that there are multiple paths at this point, and then one path is chosen. This keeps happening until either T is found, or there are no more “.” to move to. If there are no more places to move to, then you will need to back track until a multiple paths marker is found. If one of these markers is found, you will need to explore the alternate paths. If all alternate paths are explored and no treasure is found, then the treasure has no possible path to it. |

### SUMMARY: A brief summary description of the design and implementation, including how much your initial design plan evolved, the final result you achieved and the amount of time you spent as a programmer in accomplishing these results. This should be no more than two paragraphs.

|  |
| --- |
| Most of my first design’s basic ideas stayed true. The things that changed were the marker system for multiple paths, the stopping after one treasure was found, and I added the displaying of winning paths. My program takes a file line by line and makes a nested list which is the map. It then traverses the map going in the order of checking possible directions of N, S, E, and W in that order. Each time it moves to a new spot the coordinates are added to a class specific list that is in the explorer class I made for this. This list is a memory log of all of the coordinates we have checked. If we reach a place where we can no longer go in any direction the program backtracks until a spot is found where one of the surrounding directions has a path we have not explored. If a treasure is found, the path is printed and then the program continues running until all paths are exhausted. I spent about 7 hours on this programmer on this. |

### IMPLEMENTATION: A list in bullet form of specifically what was accomplished including any challenges overcome and innovations that were not specifically required by the assignment.

|  |
| --- |
| * The ability to check all of the spaces of the map for treasure * The ability to backtrack through the spaces that have been visited * The ability to print the paths that lead to treasure as a map and as directions |

### TESTING: A list in bulleted form of all input values used for testing. Here you should be careful to select representative input cases, including both representative typical cases as well as extreme cases.

|  |
| --- |
| * cave\_sample.txt was used as a test for 2 reachable treasures * cave\_1 was used as a test for if no treasure is found * cave\_2 was used as a test for size limits * cave\_3 was used as a test for large amounts of treasure that could be reached |

* FILES: A list in bulleted form of the names of all files submitted as part of this lab. Be sure to include your files as input to your program.

|  |
| --- |
| * Cave Driver.py * Stack.py * Explorer.py * cave\_sample.txt * cave\_1.txt * cave\_2.txt * cave\_3.txt |

### ERRORS: A list in bulleted form of all known errors and deficiencies

|  |
| --- |
|  |

### COMMENTS: A paragraph or so of your own comments on and reactions to the lab.

|  |
| --- |
| I thought this was an interesting and cool task. It can get confusing sometimes when you involve the backtracking and multiple paths, but overall it wasn’t too difficult. I thought there were clear instructions on what was wanted and pretty nice explanations of the overall idea. |

### DATA STRUCTURES AND ALGORITHMS: Include details below about each of data structures you decided to use in this lab and why you chose them. Also discuss how that choice informed the algorithms you used in order to solve the cave exploration problem.

|  |
| --- |
| I used nested lists and stacks. I used the nested lists for the holding of the general map and the winning paths map. I chose this because I needed to manipulate the data in both maps and the nested format allows me to arrange the data in a matrix fashion. I used stacks for both the current path and the winning paths. Using a stack allowed me to keep a log of directions I went and access them in a backtrack fashion using the pop() method. Since I would be using a stack I knew that each time I popped off a direction I would need to check if I could go in a different direction. This affected how I solved the problem. |