**NTUST OOP Midterm Problem Design**

|  |
| --- |
| **Subject:** **Simple Maze Traversal** |
| **Author: 劉穎龍** |
| **Main testing concept:** Array   |  |  | | --- | --- | | **Basics** | **Functions** | | ■ C++ BASICS  ■ FLOW OF CONTROL  □ FUNCTION BASICS  □ PARAMETERS AND OVERLOADING  ■ ARRAYS  □ STRUCTURES AND CLASSES  □ CONSTRUCTORS AND OTHER TOOLS  □ OPERATOR OVERLOADING, FRIENDS,AND REFERENCES  □ STRINGS  □ POINTERS AND DYNAMIC ARRAYS | □ SEPARATE COMPILATION AND NAMESPACES  □ STREAMS AND FILE I/O  □ RECURSION  □ INHERITANCE  □ POLYMORPHISM AND VIRTUAL FUNCTIONS  □ TEMPLATES  □ LINKED DATA STRUCTURES  □ EXCEPTION HANDLING  □ STANDARD TEMPLATE LIBRARY  □ PATTERNS AND UML | |
| **Description:**  For a given maze, there's a simple algorithm for walking through the maze that guarantees finding the exit (assuming there's an exit). If there's no exit, you'll arrive at the starting location again.  Place your **left** hand on the wall to your **left** and begin walking forward. Never remove your hand from the wall. If the maze turns **left**, you follow the wall to the **left**. As long as you do not remove your hand from the wall. Eventually, you'll arrive at the exit of the maze. You can also use your **right** hand in the similar manner to find the exit of the maze, just use your **right** hand, follow the wall of your **right**, and turn **right** instead.   |  |  | | --- | --- | |  |  | | left hand algorithm | right hand algorithm |   **Input:**  Each test case will only contain one set of data. The input consists of integers **m specified the hand using in the algorithm**, 0 means left hand, 1 otherwise. Then follow by a integer **n and** a **n** \* **n** matrix of characters (**n** rows separated by line break, and each row has **n** characters) that represent the maze. The maze consists of ‘**X**’ and ‘**#**’, ‘**X**’ means the passage, and ‘**#**’ means the wall. The entrance cell is the **only** passage in the leftmost column, and the exit is the single passage in the rightmost column.  input constrains (you can assume all inputs have these properties):  **m** = 0 or 1, 3 <= **n** <= 20,  **maze** will only have one passage in the leftmost column, and the top and bottom row only have walls.  **Output:**  Please **print the maze**, and **number of steps** to find the exit, if there is no path to exit exists print -1.  **print the maze**: if using left hand to traverse the maze, put the character ’L’ on the path you had taken, put the character ‘R’ if you use right hand.  **Number of steps**: only moving between cell consider a step, you’re already in the entrance cell at the start.  **Sample Input / Output：**   |  |  | | --- | --- | | Sample Input | Sample Output | | Test Case 1:  1  5  #####  #XXX#  XX###  #XXX#  #####  Test Case 2:  1  5  #####  #XXXX  XX###  #XXX#  #####  Test Case 3:  0  5  #####  #XXXX  XX###  #XXX#  ##### | Test Output 1:  #####  #RRR#  RR###  #RRR#  #####  -1  Test Output 2:  #####  #RRRR  RR###  #RRR#  #####  11  Test Output 3:  #####  #LLLL  LL###  #XXX#  #####  5 | |
| **□ Eazy,Only basic programming syntax and structure are required.**  **■ Medium,Multiple programming grammars and structures are required.**  **□ Hard,Need to use multiple program structures or more complex data types.** |
| **Expected solving time:**  45 minutes |
| **Other notes:** |