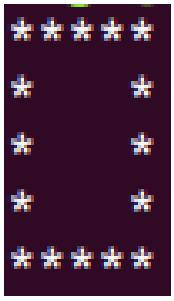
Task: For this assignment, your job is to create the maze game according to given specifications.

The objective of the player in this game is to find their way out of the maze before coming across with an enemy.

These are the rules:

- The maximum allowed board size is 50x50.
- The effective board size will be provided at runtime with the given input. In the example maze below, the dimensions of the maze is given as 5x5. For a given maze size, the outer edges will always be set as the wall (Note that '*' characters are used to represent the walls).



- In the input format, maze content will be provided after the maze size (Refer to the maze input file format at the end of this document). Maze content will be displayed according to the input specifications. In input may contain;
    - 's' represents the start position: <s>[WS]<row_number>[WS]<col_number> : The user's initial position in the maze
    - 'f' represents the final position: <f>[WS]<row_number>[WS]<col_number> : The user's final position in the maze
    - 'b' represents a barrier position: <b>[WS]<row_number>[WS]<col_number>
    - '$' represents a bonus position and the amount of the bonus is also specified as: <$>[WS]<row_number>[WS]<col_number>[WS]<amount>
    - 'X' represents an enemy position: <X>[WS]<row_number>[WS]<col_number>
    - 'end' represents that the input specification for the maze is

  ended. where [WS] is the white character (space).

- In this state user can traverse within the maze using characters ‘d’,’r’,’l’,’u’. In each step, you have to print the maze content and show the new position of the player. (Please pay attention to the sample output formats for printing the maze contents) At any stage, if user tries to move through a ‘wall’ or a ‘barrier’, your program will not make that movement and prompt: **“INVALID MOVE. TRY ANOTHER DIRECTION!”;**in this case the maze is printed again. Each movement (either valid or invalid) is always shown on the screen.
    - ‘d’ is the command to go ‘down’ for the player
    - ‘r’ is the command to go ‘right’ for the player
    - ‘l’ is the command to go ‘left’ for the player
    - ‘u’ is the command to go ‘up’ for the player
    - ‘q’ is the command to quit from maze game.

- **Input format:**

  <row>[WS]<column>

  <s><[WS]><row_number>[WS]<col_number>

  <f><[WS]><row_number>[WS]<col_number>

  (<b | $ | X>[WS]><row_number>[WS]<col_number>)*

  end

  <d | u | r | l | q>*

  where [WS] is the white character (space), *: zero or more, |: or

- There are 3 termination conditions in this game:
    - If the user successfully reaches the final position, the game is over and your program prompts: **“CONGRATS! YOU GOT %d BONUS :)\n”** then print the final state of the maze.
    - If the user meets with an enemy, he/she dies and in this case your program prompts: **“YOU MET WITH THE ENEMY AND LOST THE GAME :(\nGAME IS OVER!\n”** then you print the final state of the maze.
    - If the user quits the maze game **or** all the input commands are consumed before reaching the final position, the game is over and your program prompts: **“YOU COULD NOT REACH THE FINAL POSITION :(\nGAME IS OVER!\n”** then print the final state of the maze.

Warning: Any form of code copying, including the copies from the internet, is strictly prohibited. If we determine similarities between your codes with any other students in the class, it will be treated as cheating and will be punished. So, you would increase the risk of cheating when you see somebody else's code directly or see a solution in the internet (i.e. somebody else might have also copied the same code from the internet like you, so both of these codes will be evaluated as copies, since they both copy from an external source. Such attempts will always be considered as cheating). You are supposed to write the program by yourselves.

Testing:

As usual, use *input redirection* mechanism of your operating system to test your programs. For example, if your executable is called as PA2, redirect the input.txt file to standard input using **<** operator and redirect your outputs to a file using **>** operator such as:

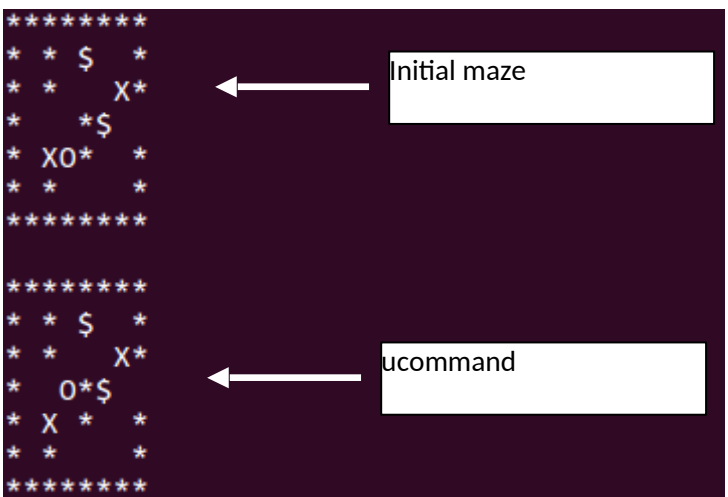
> ./PA2<input1.txt>myOutput1.txt

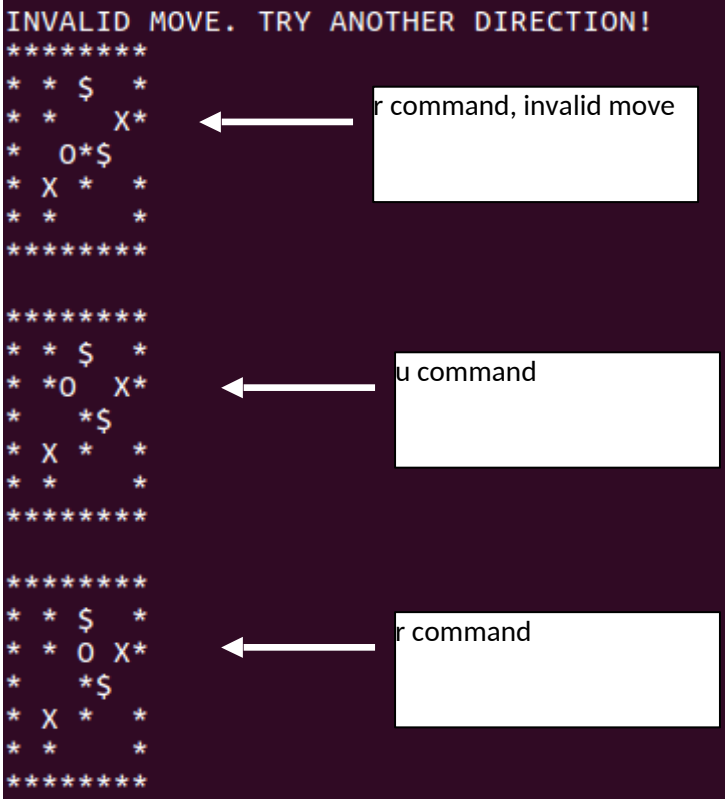
Then compare your outputs file with the given output files such as:
>diff myOutput1.txt output1.txt

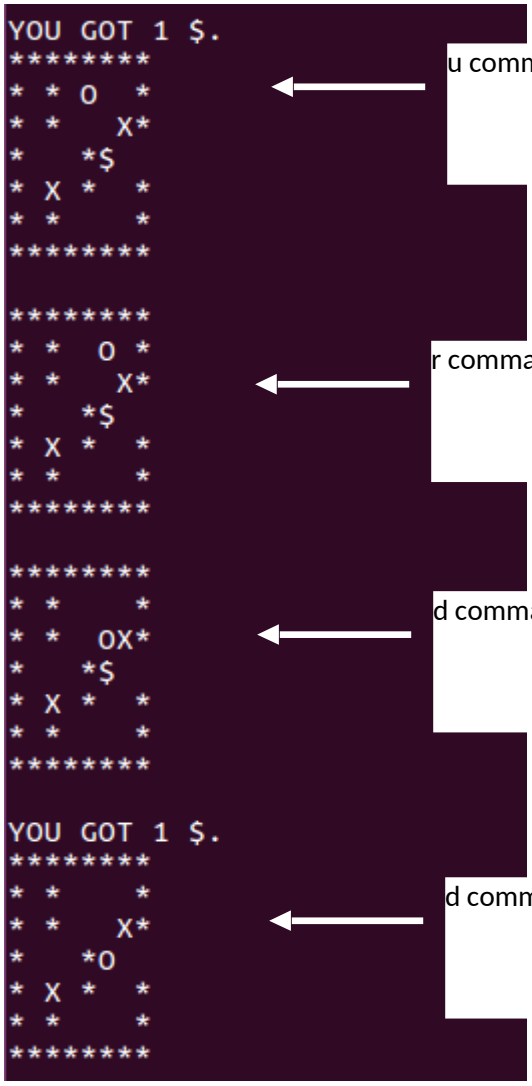
Repeat these steps for all the given input and output files.
This kind of execution enables your programs to read inputs from a file without writing any file related functions (e.g. fopen(), fscanf() etc.). In other words, the getchar() or scanf() functions in your code reads data from the redirected files instead of the std. input in this way (e.g. keyboard).

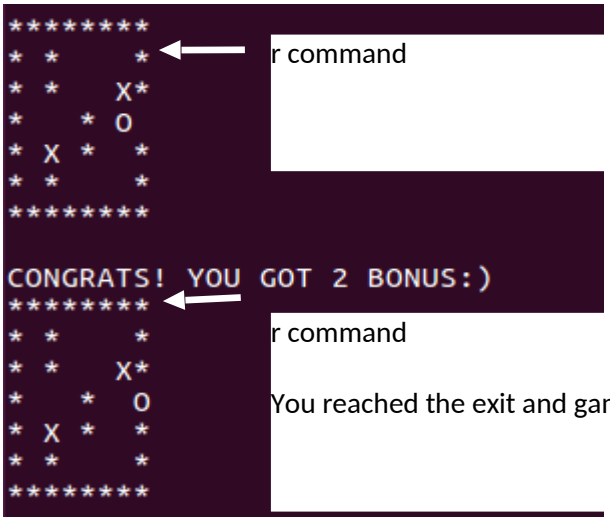
Example: This example is also given to you with the I/O files as input3.txt and output3.txt.

7 8 // maze size
s 4 3 // start position
f 3 7 // finish position
$ 2 2 // prize position
b 2 2 // barrier position
$ 1 4 1
b 5 2
X 2 6 // enemy position
b 3 4
$ 3 5 1
X 4 2
b 1 2
b 4 4
end
urururddrr // commands




INVALID MOVE. TRY ANOTHER DIRECTION!
****
* * $ *
* *   X*
* O*$
* X * *
* *   *
****
r command, invalid move
****
* * $ *
* *O X*
*   *$
* X * *
* *   *
****
u command
****
* * $ *
* * O X*
*   *$
* X * *
* *   *
****
r command


YOU GOT 1 $.
****
* * O *
* *   X*
*   *$
* X * *
* *   *
****
u command, you got 1 $. Pay attention to $ symbol is deleted!
****
* * O *
* *   X*
*   *$
* X * *
* *   *
****
r command
****
* *   *
* * OX*
*   *$
* X * *
* *   *
****
d command
YOU GOT 1 $.
****
* *   *
* *   X*
*   *O
* X * *
* *   *
****
d command, you got 1 $. Pay attention to $ symbol is deleted!



r command

r command

You reached the exit and game is over.