COMS 3136 Essential Data Structures in C/C++, Lab #5 (100 points total)

Please read this assignment carefully and follow the instructions EXACTLY.

Submission

For this lab, do NOT create part1 or part2 directory. You will add the functionalities of both parts into the same code. After you read this instruction completely, you can choose to work on the two parts in any order.

Please refer to the lab retrieval and submission instruction, which outlines the only way to submit your lab assignments. Please do not email me your code.

Please make sure that your submission satisfies the requirements for the following items:

- README.txt
- Makefile
- Valgrind

The requirements remain the same as lab 1.

Part 1: Maze solver (60 points)

The skeleton code for this lab is a fully functional maze game. Here is the screenshot:

```
########################################
#.....#.......#         #              #
#####.#.#####.# ####### # ########### #
#   #.#.#   #.#     #   #   #       # #
# # #.#.### #.##### # ##### # ### ### #
# # #.#...#...#...D # #     #   # #   #
# # #.###.#.###.##### # ##### # # # # #
# # #.#...#.....#     #   #   # # # # #
# # #.#.######### ####### ##### # # # #
# # #.#...#     #   #           # # # #
# ###.###.# ### ### ############### # #
#    .....#   #                     #G#
########################################
```

The objective is to move "the dude" (’D’ in the maze) using the arrow keys on your keyboard to navigate the maze to find the gold (’G’ in the lower right corner).

Try it by typing "make" and then "./maze". You can press ’q’ or Ctrl-C to quit if you get bored. The program will start by randomly generating a maze as big as your terminal screen. (So if you want to actually solve the maze, you may want to make your terminal screen small.)

Before you can tackle the task I ask in this part, you need to study the source code I gave you. It uses the Ncurses library to manipulate the terminal screen in a way we have not seen before, but the code is heavily commented, so you should be able to figure out how it works by carefully following the code. You can find a lot of information, documentations and tutorials on the Internet, but learning how to use it by studying the maze source code should be sufficient for doing your assignment.

You job is to modify the code to provide the "auto solve" feature. I put the executable built from my solution in /home/jae/cs3136-pub/bin directory. Try running that executable. At any point in the game, you can press 'a' to activate the auto solve. From that point, the dude will start moving on its own (from the spot he was when you pressed 'a') until he finds the gold, at which point the game will end. Here is how it looks like:

```
########################################
#     #       #.........#.............#
##### # ##### #.#######.#.###########.#
#   # # #   # #.....#...#...#.......#.#
# # # # ### # #####.#.#####.#.###.###.#
# # # #   #   #   ..#.#.....#...#.#...#
# # # ### # ### #####.#.#####.#.#.#.#.#
# # # #   #     #.....#...#...#.#.#.#.#
# # # # #########.#######.#####.#.#.#.#
# # # #   #     #...#...........#.#.#.#
# ### ### # ### ###.###############.#.#
#         #   #    .................#D#
###############[GAME OVER]##############
```

Some tips and requirements:

- The function build_maze(), which in turn calls a recursive dig_maze(), populates a char matrix (vector< vector<char> >) with walls (represented as '#') and corridors (represented as ' '). It is not essential for you to understand completely how it works for doing your assignment, so you may skim that part first. However, understanding the maze creation algorithm may help you figure out how to solve the maze. The maze creation is basically a DFS. This Wikipedia article explains the algorithm: http://en.wikipedia.org/wiki/Maze_generation_algorithm

- The random number generator is seeded with the screen dimension, so the same maze will be generated on the same-size screens. This will be useful if you want to compare the behaviors of your solution with those of mine.

- The most important objective here is to implement the auto solve that works, of course. That is, the dude will find its way to the gold (you need to be able to see it doing it), and then the dude will stop moving, ending the game.

- You are required to pause for 100 milliseconds after each move so you can watch the dude's movement. In fact, let me give you a few lines from my solution that you may find useful:

```
// Draw the maze, the gold, and the dude.
// Note that the gold and the dude are not stored in the
// maze matrix; they are just drawn on top of the maze.
display_maze(m);
```

```
        mvaddch(goldRow, goldCol, MAZE_GOLD);
        mvaddch(r, c, MAZE_DUDE);
        refresh();
        usleep(DELAY); // wait a little bit.
```

    DELAY is defined as:

```
        const int DELAY = 100 * 1000;
```

- Once you get the core functionality working, you can then try to make your auto solve behave as closely as possible to my version. In order to get full credit, yours need to behave exactly the same as mine.

  Here are some observations on the behavior of auto-solve:

  - Given a choice between multiple directions it can take, the dude will choose the direction in the following order: north, east, west, south (you can remember it as NEWS).

  - The dude will always move continuously, i.e., it does not jump to a non-adjacent cell. You can see that, when he's cornered and cannot proceed, he backtracks smoothly.

  - '.' is displayed on a cell that the dude has visited. 'D' and 'G' are visible throughout the game, except at the end when they are at the same location, 'D' will be displayed.

  - The dude will start moving from where he was when you pressed 'a'.

Part 2: Forget gold, I got a briefcase full of cash! (40 points)

Try running my executable again, and at any point in the game (but not during auto-solve), press 'f' to activate "flooding":

Flooding in progress:

```
##################################
#.#$$$$$$$#                    ##
#.#$#####$####### ##### ##### ##
#.#$$$$$#$$$$$$$# # #   #   # ##
#.#####$#######$# # # ##### # ##
#.#$$$#$#$$$#$$$#   # #   # # ##
#$#$#$#$#$####$####### # #$# # ##
#$$$#$$$#$#$$$#$$$$$#   #$# # ##
#########$#$###$#$#$####$# # ##
#$#$$$$$$$#$#$$$#$#$$$$$#$# # ##
#$#$###$###$#####$#####$#$# # ##
#$$$$$#$$$$$$$$$$#$$$$$$$#  G##
##################################
```

Flooding done:

```
##################################
#$#$$$$$$$#$$$$$$$$$$$$$$$$$$##
#$#$#####$#######$#####$#####$##
#$#$$$$$#$$$$$$$#$#$#$$$#$$$#$##
#$#####$#######$#$#$#$#####$#$##
#$#$$$#$#$$$#$$$#$$$#$#$$$#$#$##
#$#$#$#$#$####$#######$#$#$#$#$##
#$$$#$$$#$#$$$#$$$$$#$$$#$#$#$##
#########$#$###$#$#$####$#$#$##
#$#$$$$$$$#$#$$$#$#$$$$$#$#$#$##
#$#$###$###$#####$#####$#$#$#$##
#$$$$$#$$$$$$$$$$#$$$$$$$#$$$##
##########[GAME OVER]###########
```

The dude, due to the stress of chasing gold for so long, will go crazy and start throwing money he had in his briefcase until the corridors are filled with '$'.

At every junction, he even clones himself so that he can move on to all possible directions at the same time. (It's actually not happening at the

same time, but it kinda looks that way.)

Your job is to implement this flooding. You may find that flooding is actually easier than solving the maze. You can do part 2 before part 1 in that case.

--

The backdrop for this lab is inspired by the movie, The Big Lebowski (1998), with Jeff Bridges playing "the dude". It’s one of the funniest movies I’ve ever watched.

Good luck and enjoy!