CSC 112 Lab 8
Rat Maze
Fall 2007

Due
3:00pm Friday, 11/9

1 Program Description

Dr. Pluf loves to watch the classic experiment of a rat in a maze. The rat starts at one end of the maze and traces paths through the maze searching for cheese at the end of the maze. Dr. Pluf just can't witness enough of these experiments, so he asks you, to write a program that simulates this experiment. This way Dr. Pluf can watch an experiment any time he wants to using his PC running Linux. Sadly, Dr. Pluf cannot afford a Mac...

To simulate the experiments you will represent a maze as a fixed dimension 2D array filled with `0`'s and `1`'s. Let the `1`'s represent barriers (a maze wall) and the `0`'s valid locations that the simulated rat can occupy. The rat can move to any contiguous (horizontally, vertically, or diagonally) `0` (space). Your job is to design and implement a program that finds **the** path from the start to the end (using the algorithm described below), providing such a path exists. The program will first prompt the user for a *random seed number*, then randomly generate a maze. Afterwards, the original maze is displayed and the path is determined. The output of your program should contain a graphical rendition of the maze containing `0`'s and `1`'s. If a path from the start to the end exists, display another rendition of the maze, but this time include `*` for each step in the path; however, do **not** include any side trips. If the maze does not contain a path, output the message "`No path exists!`"

1.1 Determining the Path

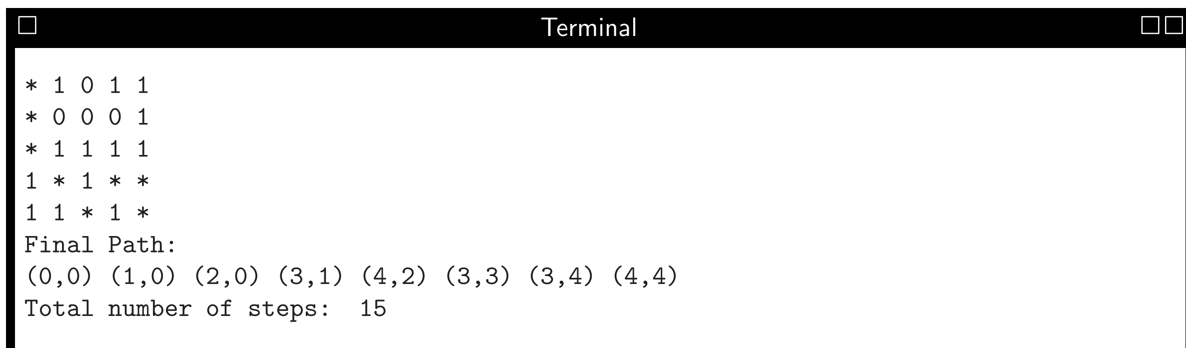
The maze entrance is **always** the top left corner and the maze exit is **always** the bottom right corner.

```
0 1 0 1 1
0 0 0 0 1
0 1 1 1 1
1 0 1 0 0
1 1 0 1 0
```

Let the top left corner of the maze be location `(0, 0)`. Now if possible, the rat takes another step in one of eight directions, where the eight directions from a location x are

4	3	2
5	x	1
6	7	8

Therefore, the rat will *first* try to move to the east. If the rat has already tried this space or if it is a maze wall, then the rate tries the north-east direction, etc... The rat repeats the process until the exit is reached or the rat is at a *dead-end* (nowhere to move). In the latter case, the rat moves to a previous position with an unexplored step and continues finding a path. If a path exists, output the final path graphically, substituting '`*`' for steps along the path and display the list of step locations in the path in (row, col) format with no more than 10 steps per line (again, do **not** include an side trips). In addition, print the **total** number of steps the rat made (including all side trips).



2 Program Requirements

You **must** adhere to all of the following points to receive credit for this program.

- Turn-in (print-outs and electronically) the files for this program.
- Program must be modular and consist of the following 4 files
  1. `main.cpp` contains the main function which reads the seed value and calls functions to generate a maze, determine the path, and print the maze
  2. `maze.h` contains the declarations for your maze functions
  3. `maze.cpp` contains the definitions for your maze functions
  4. `makefile` a makefile that compiles your code, where the compilation generates object files then links for a final executable called `maze`
- Define the maze dimensions as 25×25
- Prompts and reads a *random number seed* from the user, then randomly populate the maze with `0`'s and `1`'s
- Search for a path from the start location `(0, 0)` to the end location `(24, 24)`. Allow the rat to back up if necessary.
- Print the maze with `*` for locations along the path and a listing if it exists.
- Print the "`No path exists`" message if a path does not exist
- Always print the total number of steps made.
- Program must adhere to documentation style and standards.

3 Programming Hints

- Maze generator is available at the course web-site
- Use `#defines` for sizing your arrays
- Your first attempt should only search in a *forward* direction, add *backing-up* later
- Test with small-sized (4×4) mazes