Rebecca Lee
013919802
Kayte Chien
014217970

Lab 3 Report

The maze generation algorithm we implemented was Prim's algorithm. We generated a matrix of nodes and chose a random one to be the starting point, then added its neighbors to the frontier. The full list of instructions are as follow (also commented in the code):

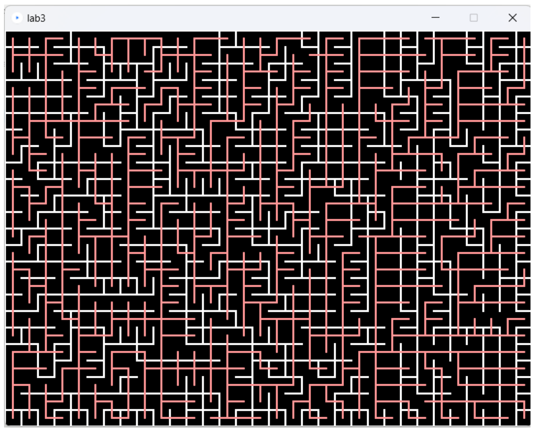
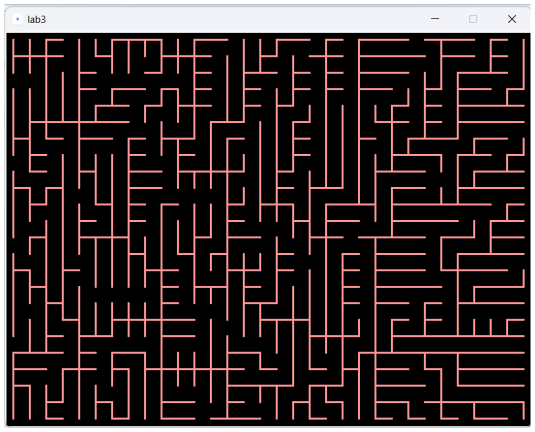
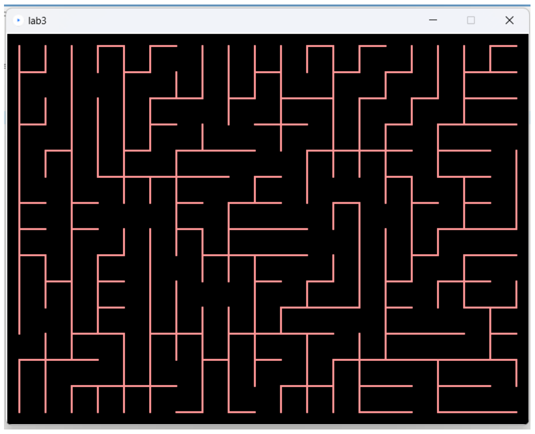
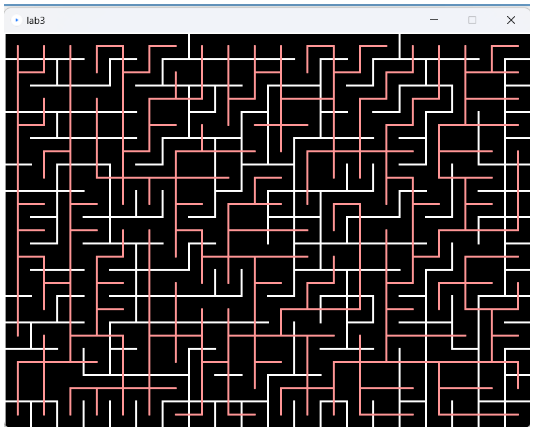
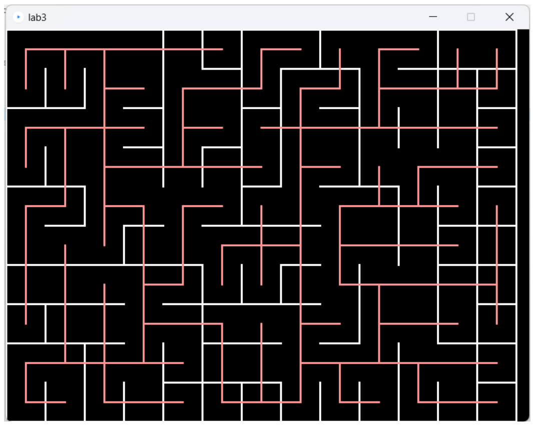
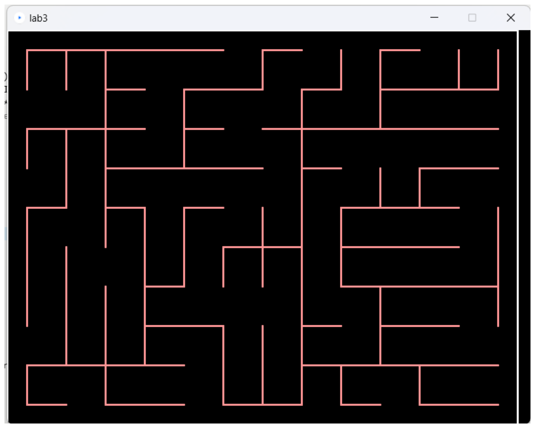
1. Pick a random node
2. Add the point to “the maze” (the tree)
3. Add the point’s neighbors to the frontier

```
while(frontier.size() > 0)
```

4. Pick a random node on the frontier
5. Connect node to adjacent node in the tree (i.e. set as neighbors)
6. Add the random node’s neighbors to the frontier
7. Add random node to the tree (i.e. mark as visited, remove from frontier)

The program may add a node to the frontier multiple times (and a node on the frontier may have been added to the tree), so before any operations are done (i.e. steps 5-7), the program checks if the node was previously visited.

The bottom and right walls of a node were stored in the MazeCell class, and when a neighbor was added, the wall between the node and its neighbor was removed. This method was met with varying degrees of success (i.e. correctly removed some walls, incorrectly removed others).

Grid size	Maze with walls	Maze without walls
25		
40		
60		

150