Final Programming Assignment
Carson Hanel

Preface: Alright, again, I realize this isn't LyX but I thought that this would be just as good given that there is no template meant to be used for the report.

1. Assignment description:
    a. The assignment was to create a program which read in a maze of a perfect square, created a graph based upon an adjacency list, solved the maze from the top left corner to the bottom right corner, and if possible gives the solution to the user.
2. Description of DS&A used:
    a. The data structures used include:
        i. The class adjacency node
            1. Used to house adjacent values as nodes in an adjacency list.
            2. Has two variables: ID and a pointer to the next node.
        ii. The class adjacency list
            1. Used to store the head node of an adjacency list.
        iii. The STL list (utilizing as a queue)
            1. Used to store vertices within the BFS algorithm.
            2. Operations used on the queue are push_back() and pop()
        iv. The STL list (utilizing as a buffer. I guess also queue.)
            1. Used to hold the values of the southern gates of vertices in graph printing.
        v. The class Graph
            1. Holds an adjacency list, number of vertices, number of edges, and a 2D matrix of vertices that is really unnecessary, but I implemented it anyways.
        vi. The class Vertex (simply for data representation. Not necessary.)
            1. Holds its integer ID, and four pointers to the four cardinal directions for graph connectivity. Null pointers are seen as walls, and pointers pointing to adjacent vertices are doors.
    b. The algorithms used include:
        i. Breadth-First Search used to find the path from the node at 0 to the nth vertex. The asymptotic complexity of BFS is O((m+n)log(n))
        ii. A Breadth-First search helper function that prints out the path if one is found. The asymptotic complexity of this function is O(m)
3. A user guide to navigate my program:
    a. Simply, you can open the program (exe) and type in the file name. The rest is done for you.
4. Testing:
    a. So, some of the tests I implemented were pretty obvious.

b. If there are more direction values to be read in from the given file, simply the program truncates the extra data and attempts to find a solution.
c. If there are less direction values to be read in from the given file, the program issues a warning to the user, the program is paused, and upon entry of a character terminates.
d. If there is no path to an end-node, the BFS terminates, issues a warning to the user, and upon entry of a character terminates.
e. If the size variable sent to the function at the beginning of the file is not a perfect square, do not process the input, issue a warning to the user, and upon entry of a character terminates the program.
f. All of these cases are implemented in my program.