```h
//
// Created by abelg on 2023-02-14.
//

#ifndef ASSIGNMENT_2_MAZE_H
#define ASSIGNMENT_2_MAZE_H

#include <vector>
#include "Stack.h"

class Maze {
    Stack stack;
private:
    std::vector<std::vector<char>> maze;
public:
    Maze();

    ~Maze();

    void displayMaze();

    void loadToMaze();

    void exploreMaze();

    void clearMaze();

    void markMaze();

    void writeToFile();
    //bool isNotVisited(int x, int y);
};

#endif //ASSIGNMENT_2_MAZE_H

```

```
1  //
2  // Created by abelg on 2023-02-14.
3  //
4
5  #ifndef ASSIGNMENT_2_NODE_H
6  #define ASSIGNMENT_2_NODE_H
7
8  #include "Coordinate.h"
9
10 struct Node {
11     Coordinate coordinate;
12     Node *m_next{nullptr};
13 };
14 #endif //ASSIGNMENT_2_NODE_H
15
```

```
 1 //
 2 // Created by abelg on 2023-02-14.
 3 //
 4
 5 #ifndef ASSIGNMENT_2_STACK_H
 6 #define ASSIGNMENT_2_STACK_H
 7
 8 #include <ostream>
 9 #include <vector>
10 #include "Node.h"
11
12 class Stack {
13
14 private:
15     Node *m_first{nullptr};
16 public:
17     void push(Coordinate coordinate);
18
19     void pop();
20
21     friend std::ostream &operator<<(std::ostream &
   output, Stack stack);
22
23     Coordinate peek();
24
25     std::vector<int> xCoordinates();
26
27     std::vector<int> yCoordinates();
28 };
29
30 #endif //ASSIGNMENT_2_STACK_H
31
```

```cpp
#include <iostream>
#include <fstream>
#include "Maze.h"
#include "Stack.h"


using namespace std;

int main() {
    //create maze object
    Maze maze;
    //load maze to vector
    maze.loadToMaze();
    //solve maze
    maze.exploreMaze();
    //clear marked spots
    maze.clearMaze();
    //mark maze
    maze.markMaze();
    //display maze to the console
    maze.displayMaze();
    //write maze to the file
    maze.writeToFile();

    return 0;
}
```

```cpp
//
// Created by abelg on 2023-02-14.
//
#include <vector>
#include <fstream>
#include <iostream>
#include "Maze.h"
#include "Coordinate.h"


Maze::Maze() = default;

Maze::~Maze() = default;


//load maze to double dimensional vector
void Maze::loadToMaze() {
    //declare file reader and passing the file.
    //std::ifstream  fin(R"(C:\Users\abelg\CLionProjects\assignment-2-Abel-Berhe\docs\maze.txt)");
    //std::ifstream  fin(R"(C:\Users\abelg\CLionProjects\assignment-2-Abel-Berhe\tests\test.txt)");
    //std::ifstream  fin(R"(C:\Users\abelg\CLionProjects\assignment-2-Abel-Berhe\tests\test2.txt)");
    std::ifstream fin(R"(C:\Users\abelg\CLionProjects\assignment-2-Abel-Berhe\tests\test3.txt)");
    std::string line;
    //check if file exist
    if (fin.is_open()) {
        while (getline(fin, line)) {
            //vector to store row
            std::vector<char> row;
            for (char c: line) {
                if (c != '\0') {
                    row.push_back(c);
                }
            }
            //store row in the 2D vector
```

```
35              maze.push_back(row);
36          }
37
38      } else {
39          std::cout << "File not found" << std::endl;
40      }
41      fin.close();
42  }
43
44  //SolveMaze
45  void Maze::exploreMaze() {
46      //declare coordinate object;
47      Coordinate coordinate;
48      //flag to check for exit
49      bool isExit = true;
50      //Starting point at maze
51      coordinate.x = 1;
52      coordinate.y = 0;
53      //start maze that starting point
54      maze[coordinate.x][coordinate.y] = '#';
55      //pass the starting coordinates to stack
56      stack.push({coordinate.x, coordinate.y});
57      while (isExit) {
58          //coordinates should be greater than the
    starting point
59          if (coordinate.x < 1 && coordinate.y < 0) {
60              std::cout << "Error!- Outside of the Maze
     grid" << std::endl;
61              break;
62          } else {
63              //check if left for movement
64              if (maze[coordinate.x][coordinate.y - 1
    ] == ' ' && coordinate.y > 1) {
65                  //mark the path
66                  maze[coordinate.x][coordinate.y - 1
    ] = '#';
67                  //add coordinates to the stack
68                  stack.push({coordinate.x, coordinate.
    y - 1});
69                  coordinate.y--;
70                  //check bottom for movement
```

```
71              } else if (maze[coordinate.x + 1][
   coordinate.y] == ' ') {
72                  //mark the path
73                  maze[coordinate.x + 1][coordinate.y
   ] = '#';
74                  //add coordinates to the stack
75                  stack.push({coordinate.x + 1,
   coordinate.y});
76                  coordinate.x++;
77                  //check right for movement
78              } else if (maze[coordinate.x][coordinate
   .y + 1] == ' ') {
79                  //mark the path
80                  maze[coordinate.x][coordinate.y + 1
   ] = '#';
81                  //add coordinates to the stack
82                  stack.push({coordinate.x, coordinate
   .y + 1});
83                  coordinate.y++;
84                  //check top for movement
85              } else if (maze[coordinate.x - 1][
   coordinate.y] == ' ') {
86                  //mark the path
87                  maze[coordinate.x - 1][coordinate.y
   ] = '#';
88                  //add coordinates to the stack
89                  stack.push({coordinate.x - 1,
   coordinate.y});
90                  coordinate.x--;
91
92              } else {
93                  //Dead end, maze loops back the same
    path
94                  //check left to move back
95                  if (maze[coordinate.x][coordinate.y
    - 1] == '#') {
96                      //mark the path with D to
   indicate it is a dead end path
97                      maze[coordinate.x][coordinate.y
   ] = 'D';
98                      //remove coordinates from stack
```

```
 99                      stack.pop();
100                      coordinate.y--;
101                      //check bottom to move back
102                  } else if (maze[coordinate.x + 1][
    coordinate.y] == '#') {
103                      //mark the path with D to
    indicate it is a dead end path
104                      maze[coordinate.x][coordinate.y
    ] = 'D';
105                      //remove coordinates from the
    stack
106                      stack.pop();
107                      coordinate.x++;
108                      //check right to move back
109                  } else if (maze[coordinate.x][
    coordinate.y + 1] == '#') {
110                      //mark the path with D to
    indicate it is a dead end path
111                      maze[coordinate.x][coordinate.y
    ] = 'D';
112                      //remove coordinates from the
    stack
113                      stack.pop();
114                      coordinate.y++;
115                      //check top to move back
116                  } else if (maze[coordinate.x - 1][
    coordinate.y] == '#') {
117                      //mark the path with D to
    indicate it is a dead end path
118                      maze[coordinate.x][coordinate.y
    ] = 'D';
119                      //remove coordinates from the
    stuck
120                      stack.pop();
121                      coordinate.x--;
122                  }
123              }
124          }
125
126          //check if the coordinates are the at end
127          if (coordinate.x == 49 && coordinate.y == 50
```

```
127 ) {
128             isExit = false;
129         }
130     }
131
132 }
133
134 //clear maze
135 void Maze::clearMaze() {
136     //loop through maze and clear any marked
    characters
137     for (int i = 0; i < maze.size(); i++) {
138         for (int j = 0; j < maze[i].size(); j++) {
139             if (maze[i][j] == 'D' || maze[i][j] ==
    '#') {
140                 maze[i][j] = ' ';
141             }
142         }
143     }
144 }
145
146 //mark maze
147 void Maze::markMaze() {
148     //vector stores x coordinates
149     std::vector<int> xNums = stack.xCoordinates();
150     //vector stores y coordinates
151     std::vector<int> yNums = stack.yCoordinates();
152     //loop through maze and mark spot with #
    character at the right coordinates using stack
153     for (int i = 0; i < yNums.size(); i++) {
154         maze[xNums[i]][yNums[i]] = '#';
155     }
156 }
157
158 //display maze in the console
159 void Maze::displayMaze() {
160     //loop through the maze and display the result
    to the console
161     for (auto &i: maze) {
162         for (char j: i) {
163             std::cout << j << "";
```

```
164         }
165         std::cout << std::endl;
166     }
167 
168 }
169 
170 //write to file
171 void Maze::writeToFile() {
172     //declare file writer
173     //std::ofstream outPut(R"(C:\Users\abelg\
    CLionProjects\assignment-2-Abel-Berhe\solved\maze.
    txt)");
174     //std::ofstream outPut(R"(C:\Users\abelg\
    CLionProjects\assignment-2-Abel-Berhe\solved\test.
    txt)");
175     //std::ofstream outPut(R"(C:\Users\abelg\
    CLionProjects\assignment-2-Abel-Berhe\solved\test1.
    txt)");
176     std::ofstream outPut(R"(C:\Users\abelg\
    CLionProjects\assignment-2-Abel-Berhe\solved\test2.
    txt)");
177     //check if file exist
178     if (outPut.is_open()) {
179         for (auto &i: maze) {
180             for (char j: i) {
181                 //write characters to  file
182                 outPut << j << "";
183             }
184             //create space after each line
185             outPut << std::endl;
186         }
187     } else {
188         std::cout << "File not found" << std::endl;
189     }
190     outPut.close();
191 }
192 
193 
194 
195 
196 
```

197
198
199
200
201
202
203
204

```cpp
//
// Created by abelg on 2023-02-14.
//

#include "Stack.h"
#include <iostream>

//add coordinates to the stack
void Stack::push(Coordinate coordinate) {
    auto node = new Node();
    node->coordinate = coordinate;
    //add the first node
    node->m_next = m_first;
    m_first = node;
}

//remove coordinates to the stack
void Stack::pop() {
    auto node = m_first;
    m_first = m_first ? m_first->m_next : nullptr;
    delete node;
}

std::ostream &operator<<(std::ostream &output, Stack stack) {
    auto node = stack.m_first;
    while (node != nullptr) {
        std::cout << "x: " << node->coordinate.x << ", ";
        std::cout << "y: " << node->coordinate.y << std::endl;
        node = node->m_next;
    }
    return output;
}

Coordinate Stack::peek() {
    return m_first ? m_first->coordinate : Coordinate({-1, -1});
}

```

```
38 //vector to store the x coordinates
39 std::vector<int> Stack::xCoordinates() {
40     std::vector<int> numList;
41     auto node = m_first;
42     while (node != nullptr) {
43         numList.push_back(node->coordinate.x);
44         node = node->m_next;
45     }
46     return numList;
47 }
48
49 //vector to store the y coordinates
50 std::vector<int> Stack::yCoordinates() {
51     std::vector<int> numList;
52     auto node = m_first;
53     while (node != nullptr) {
54         numList.push_back(node->coordinate.y);
55         node = node->m_next;
56     }
57     return numList;
58 }
59
60
```

```
 1 //
 2 // Created by abelg on 2023-02-14.
 3 //
 4
 5 #ifndef ASSIGNMENT_2_COORDINATE_H
 6 #define ASSIGNMENT_2_COORDINATE_H
 7
 8 struct Coordinate {
 9     int x = 0;
10     int y = 0;
11 };
12
13 #endif //ASSIGNMENT_2_COORDINATE_H
14
```