```
//
//    Name: Thurlow, Taylor
// Project: 3
//       Due: 11/09/2016
//  Course: cs25602
//
//    Description:
//          A maze solver. Takes text file input using non-spaces as barriers and spaces as
//                walkable area. S denotes start, F denotes finish. Finds shortest route.
//

#include <iostream>
#include <fstream>
#include <stdlib.h>

using namespace std;

int getMaze(string filename, int maze[24][80], int &numCols) {
      ifstream mazeFile(filename);
      string str;
      int numRows = 0;
      int cols = 0;

      while (getline(mazeFile, str)) {
            str.resize(79, ' ');
            for (int i = 0; i < str.length(); i++) {
                  int newValue = str[i];
                  maze[numRows][i] = newValue;
            }

            if (str.length() > cols) cols = str.length();

            numRows++;
            if (cols == 80) break;
      }
      numCols = cols;
```

```
	return numRows;
}

void printMaze(int maze[24][80], int numRows, int numCols) {
	for (int i = 0; i < numRows; i++) {
		for (int j = 0; j < numCols; j++) {
			int theInt = maze[i][j];
			char theChar = theInt;
			cout << theChar;
		}
		cout << endl;
	}
}

bool solveMaze(int maze[24][80], string mazeName, int numRows, int numCols) {
	int startRow, startCol, finishRow, finishCol;
	bool finishReached = false;

	for (int i = 0; i < numRows; i++) {
		for (int j = 0; j < numCols; j++) {
			if (tolower(maze[i][j]) == 's') {
				startRow = i;
				startCol = j;
			} else if (tolower(maze[i][j]) == 'f') {
				finishRow = i;
				finishCol = j;
			}
		}
	}

	// Select start point
	int startPoint = 256; // starting at 256 to avoid iterator thinking a breadcrumb is
                                    // actually the finish point.
	maze[startRow][startCol] = startPoint;

	// Mark all unlabeled neighbors of points marked with step with step+1
```

```
int step = startPoint;

while(!finishReached) {
	bool spotMarked = false;
	for (int i = 0; i < numRows; i++) {
		for (int j = 0; j < numCols; j++) {
			// Executes for each cell of maze
			int mazeInt = maze[i][j];
			// Set all unlabeled neighbors to step+1
			if (mazeInt == step) {
				int testIntTop = maze[i-1][j];
				int testIntRight = maze[i][j+1];
				int testIntBottom = maze[i+1][j];
				int testIntLeft = maze[i][j-1];

				// Top
				if (testIntTop == 32) { // equal to a space
					maze[i-1][j] = step + 1;
					spotMarked = true;
				} else if (testIntTop == 70 || testIntTop == 102) { // equal to 'F' or 'f'
					maze[i-1][j] = step + 1;
					finishReached = true;
				}

				// Right
				if (testIntRight == 32) {
					maze[i][j+1] = step + 1;
					spotMarked = true;
				} else if (testIntRight == 70 || testIntRight == 102) {
					maze[i][j+1] = step + 1;
					finishReached = true;
				}

				// Bottom
				if (testIntBottom == 32) {
					maze[i+1][j] = step + 1;
```

```
                        spotMarked = true;
                    } else if (testIntBottom == 70 || testIntBottom == 102) {
                        maze[i+1][j] = step + 1;
                        finishReached = true;
                    }

                    // Left
                    if (testIntLeft == 32) {
                        maze[i][j-1] = step + 1;
                        spotMarked = true;
                    } else if (testIntLeft == 70 || testIntLeft == 102) {
                        maze[i][j-1] = step + 1;
                        finishReached = true;
                    }


                }
            }
        }

        // No more points can be reached at this point in the for loop
        if (!spotMarked && !finishReached) {
            return false;
        }

        step++;
    }

    // finish has been reached, now lets backtrack
    int lookRow = finishRow;
    int lookCol = finishCol;
    int pathLength = 0;

    maze[startRow][startCol] = 83; // 83 = 'S' in ascii
    maze[finishRow][finishCol] = 70; // 70 = 'F' in ascii
```

```
	int totalSteps = 0;

	for (int i = step; i > startPoint; i--) { // move back from step (at the finish) to the start point
		//Look in each direction for cell with value - 1
		int testIntTop = maze[lookRow-1][lookCol];
		int testIntRight = maze[lookRow][lookCol+1];
		int testIntBottom = maze[lookRow+1][lookCol];
		int testIntLeft = maze[lookRow][lookCol-1];

		if (testIntTop == i - 1) {
			lookRow -= 1;
		} else if (testIntRight == i - 1) {
			lookCol += 1;
		} else if (testIntBottom == i - 1) {
			lookRow += 1;
		} else if (testIntLeft == i - 1) {
			lookCol -= 1;
		}
		if (maze[lookRow][lookCol] != startPoint) {
			maze[lookRow][lookCol] = 46; // 46 = '.' in ascii
			totalSteps++;
		}
	}

	// Clean the maze of all trails, leaving only X, x, S, s, F, f, ., and spaces

	for (int i = 0; i < numRows; i++) {
		for (int j = 0; j < numCols; j++) {
			int theInt = maze[i][j];
			if (theInt != 32 && theInt != 88 && theInt != 120 && theInt != 83 && theInt != 70 && theInt != 46
&& theInt != 102 && theInt != 115) {
				maze[i][j] = 32;
			}
		}
	}
```

```
	cout << "Path: " << totalSteps - 1 << endl; // -1 so we don't include the start
	printMaze(maze, numRows, numCols);
	return true;
}

int main() {
	int numRows = 0, numCols = 0;
	bool solved;
	int maze[24][80];
	string mazeName;

	cout << "T. Thurlow's A-Mazing!" << endl << endl;
	cout << "Enter the maze file name? ";
	cin >> mazeName;

	numRows = getMaze(mazeName, maze, numCols);
	printMaze(maze, numRows, numCols);
	cout << endl;
	solved = solveMaze(maze, mazeName, numRows, numCols);

	if (!solved) {
		cout << "No solution." << endl;
	}

	return 0;
}
```