# Project 1

## Minesweeper Game

**CIS-5 40375**
**Hodnett, Victoria**
**February 05, 2014**

# Introduction

Minesweeper is a great game to play during those late nights on the computer, or even on those long car rides (if you have the game on a portable device).

Typically people come across this game as they browse their computer's built-in game library. Contrary to what many may believe with the first impression, Minesweeper is not a game of chance, but of strategy.

Playing Minesweeper teaches you to assess a situation before making a decision. It also teaches you to think ahead a couple of steps at a time. Each tile you choose could determine whether you'll lose instantly by uncovering a mine, or continue on by uncovering nearby tiles that give you clues as to where the mines are.

As the difficultly increases, the more important the need for accuracy is.

# Game Play

At the start of the game you will be presented a board with a number of tiles, each labeled with the letter "M". You will be prompted on which tile you would like to uncover, choosing a letter and a number (e.g. A1).

Once you make your choice, you will either uncover a mine or (a) number(s). If you uncover a mine, you lose the game. If not, the number(s) uncovered will indicate how many mines are adjacent to that particular tile. Strategize to assume which tile *is* the mine and plan to choose a tile that you declare to <u>not</u> be the tile to uncover more clues.

The game continues until you uncover a mine or you uncover all tiles excluding the mines, which will indicate that you have won.

A winning board will display an "F" (representing a flag) in place of all the mines.

| Difficulty | Grid Size | Mines |
|---|---|---|
| **Easy** | 9 x 9 | 10 |
| **Not so Easy** | 16 x 16 | 40 |

# Summary

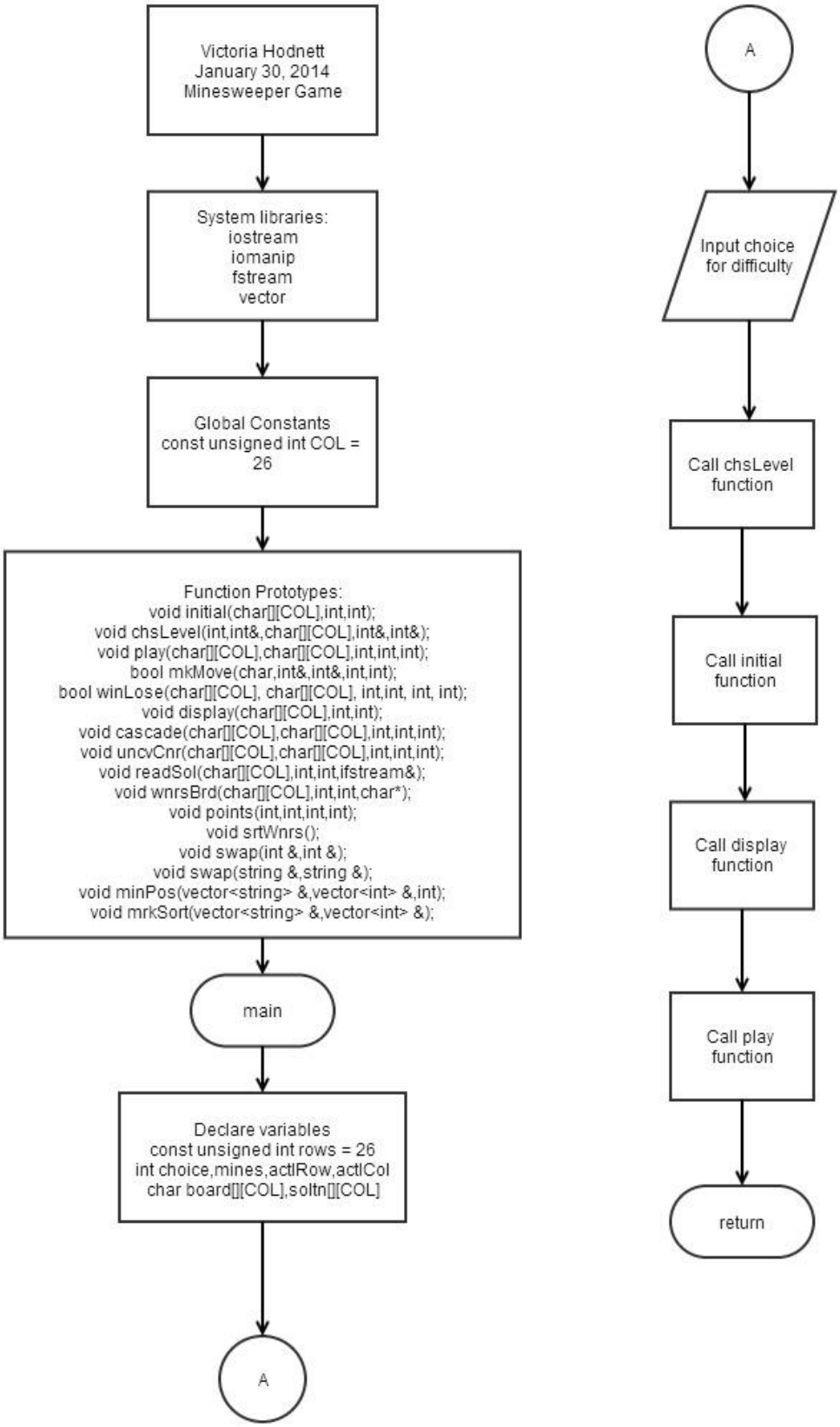
| | |
|---|---|
| **Lines of Code** | 382 |
| **Comment Lines** | 138 |
| **Blank Lines** | 22 |
| **Total Lines** | 542 |

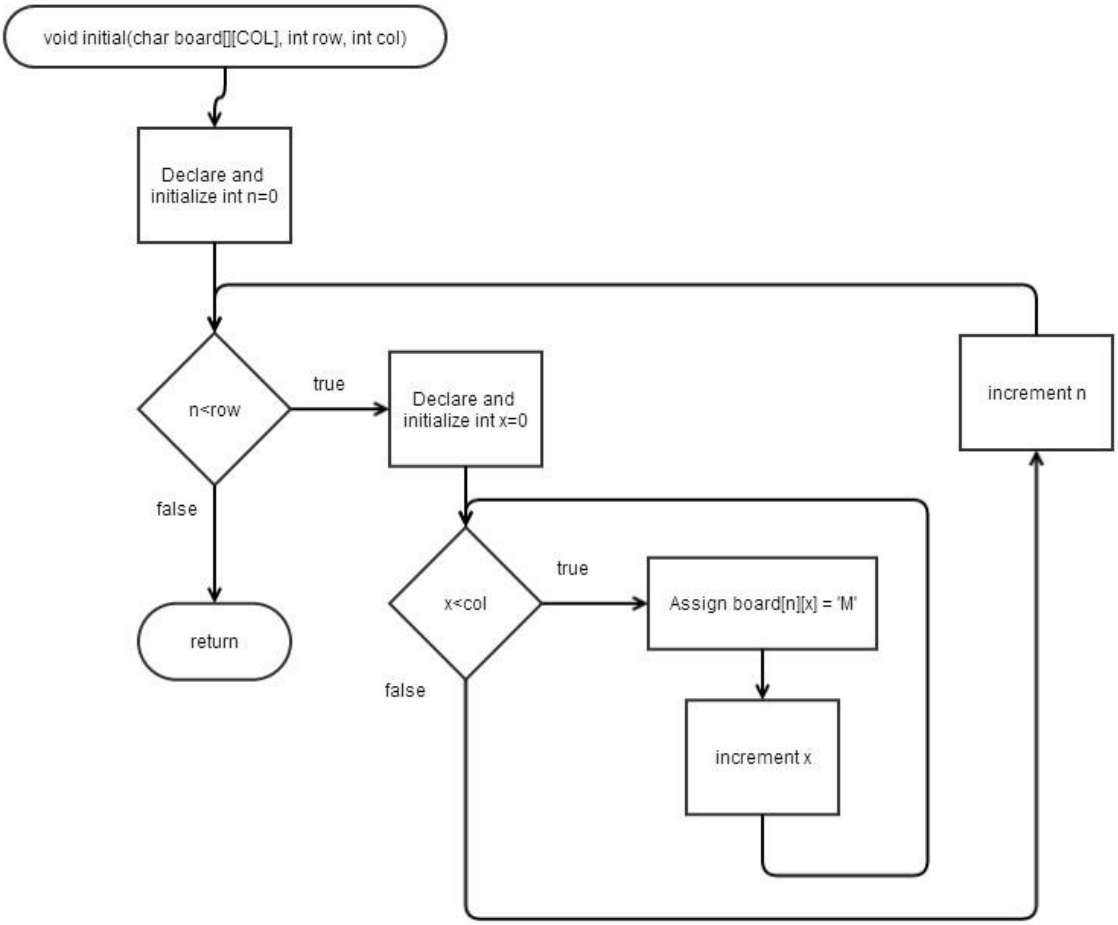
For the implementation of this program I learned how to pass a file as an argument to a function. I also learned how to append output to a file.

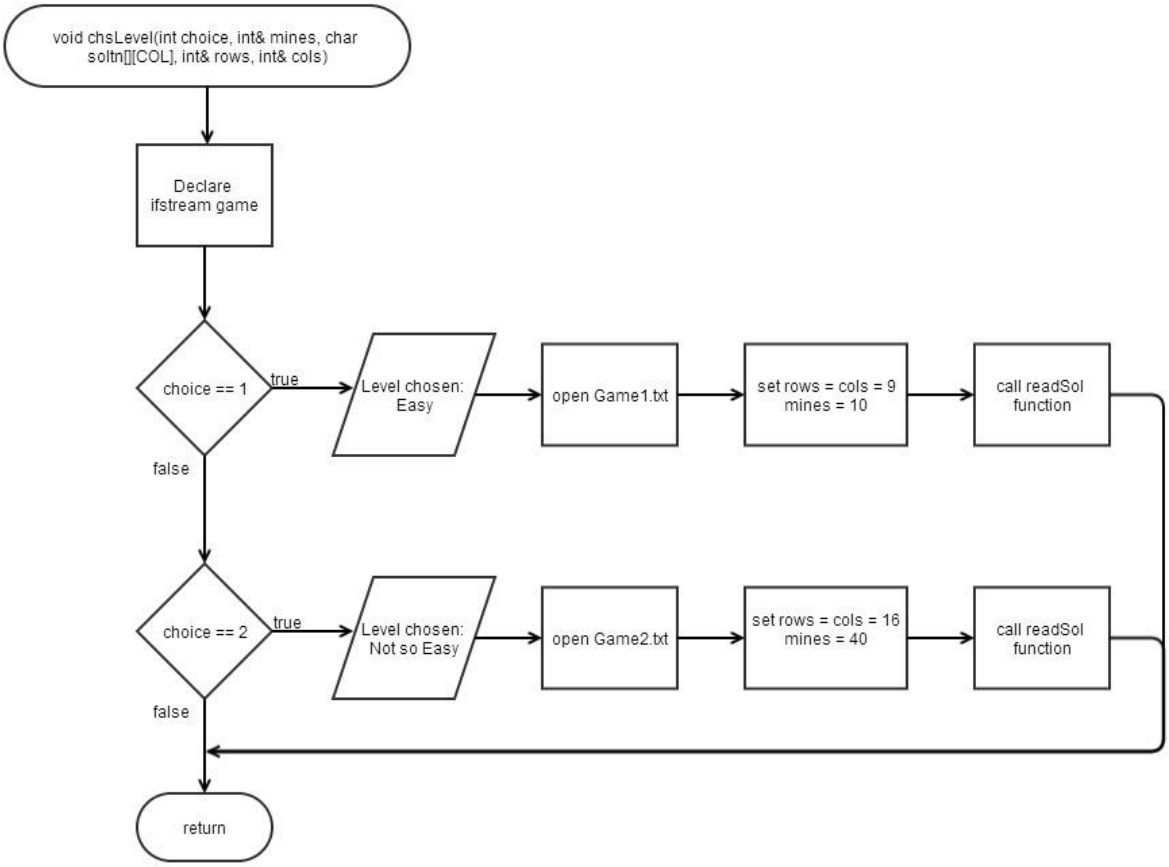
Completing the entire project took about two weeks, one for initial version, and the second to add on more constructs. The bulk of the program was very easy to implement, but I did face a problem when trying to append winners' names to a file on a new line each time. After searching in a couple of C++ forums I realized that, although the names did not appear on a new line when I opened the .txt file in Notepad, it did in Microsoft Word.

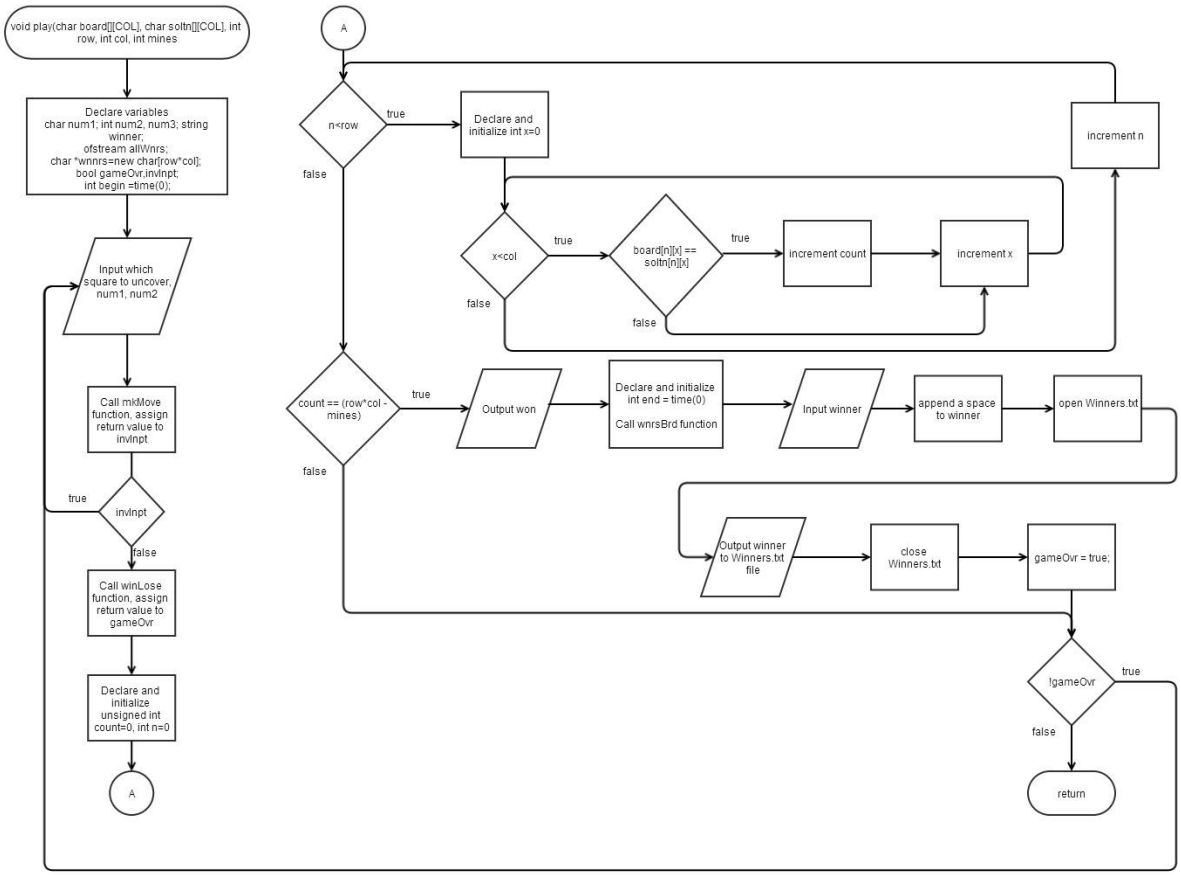
The main part of this program was to demonstrate tile selection using arrays, and to use the idea of parallel arrays (the game board and the solution board arrays go hand in hand). I also implemented sorting of arrays, by display High Scores in a file, from highest to lowest scores (up to 5). Another feature I used was the time function to award points based on how long it took to complete the level.

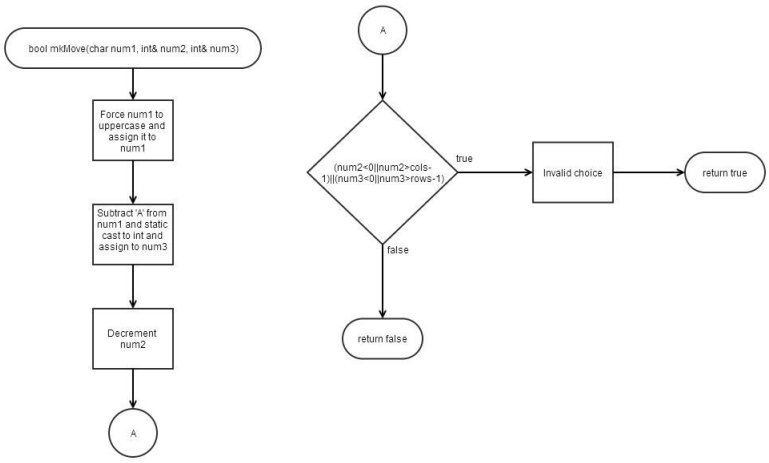
# Flowchart

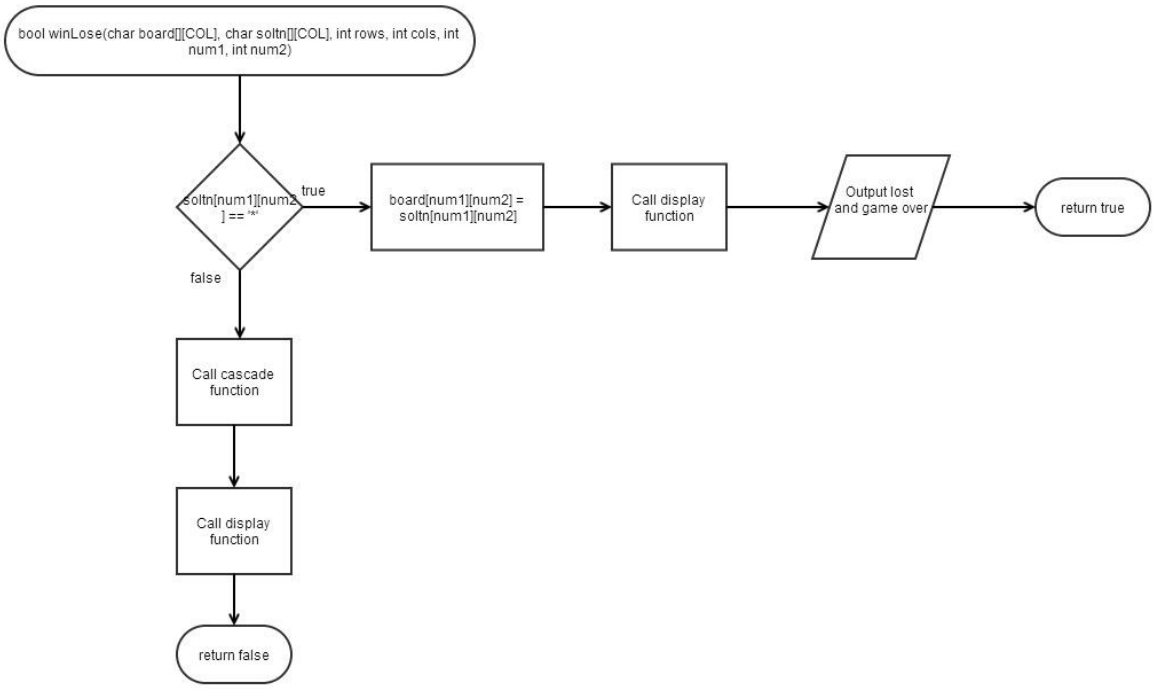

Victoria Hodnett
January 30, 2014
Minesweeper Game
System libraries:
iostream
iomanip
fstream
vector
Global Constants
const unsigned int COL = 26
Function Prototypes:
void initial(char[][COL],int,int);
void chsLevel(int,int&,char[][COL],int&,int&);
void play(char[][COL],char[][COL],int,int,int);
bool mkMove(char,int&,int&,int,int);
bool winLose(char[][COL], char[][COL], int,int, int, int);
void display(char[][COL],int,int);
void cascade(char[][COL],char[][COL],int,int,int);
void uncvCnr(char[][COL],char[][COL],int,int,int);
void readSol(char[][COL],int,int,ifstream&);
void wnrsBrd(char[][COL],int,int,char*);
void points(int,int,int,int);
void srtWnrs();
void swap(int &,int &);
void swap(string &,string &);
void minPos(vector<string> &,vector<int> &,int);
void mrkSort(vector<string> &,vector<int> &);
main
Declare variables
const unsigned int rows = 26
int choice,mines,actlRow,actlCol
char board[][COL],soltn[][COL]
A
A
Input choice for difficulty
Call chsLevel function
Call initial function
Call display function
Call play function
return

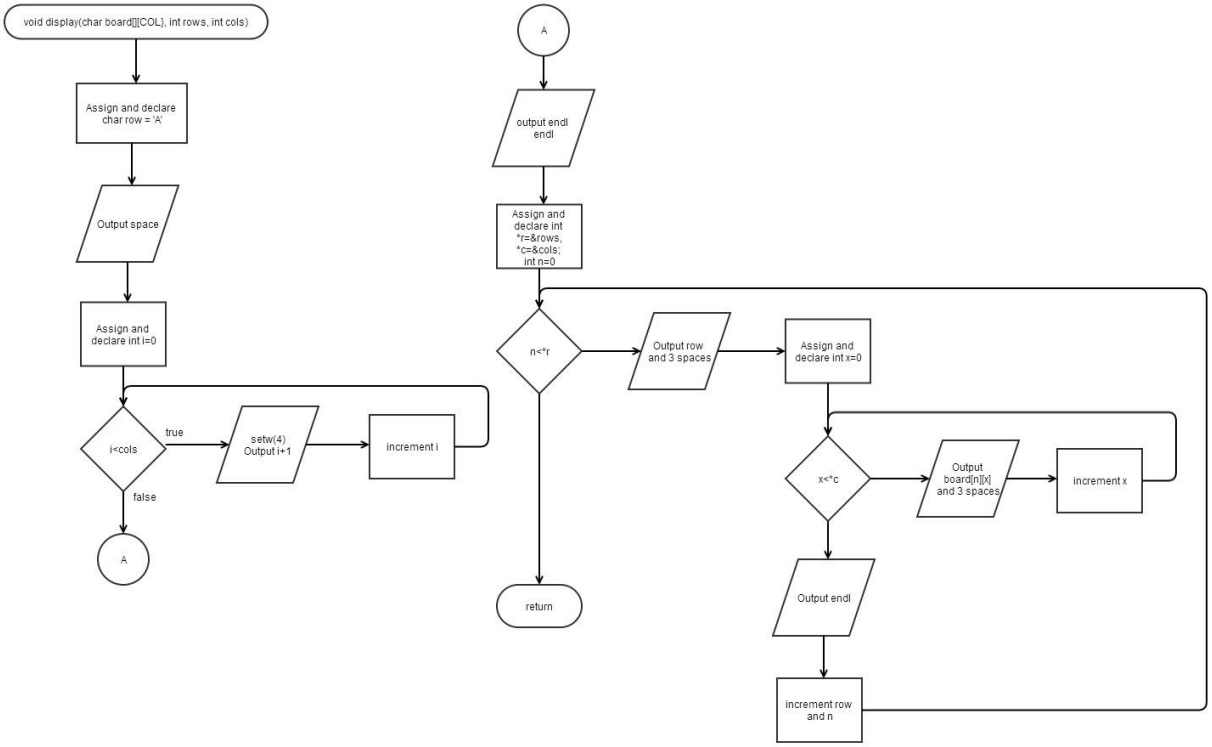

void initial(char board[][COL], int row, int col)
Declare and initialize int n=0
n<row
true
false
return
Declare and initialize int x=0
x<col
true
false
Assign board[n][x] = 'M'
increment x
increment n

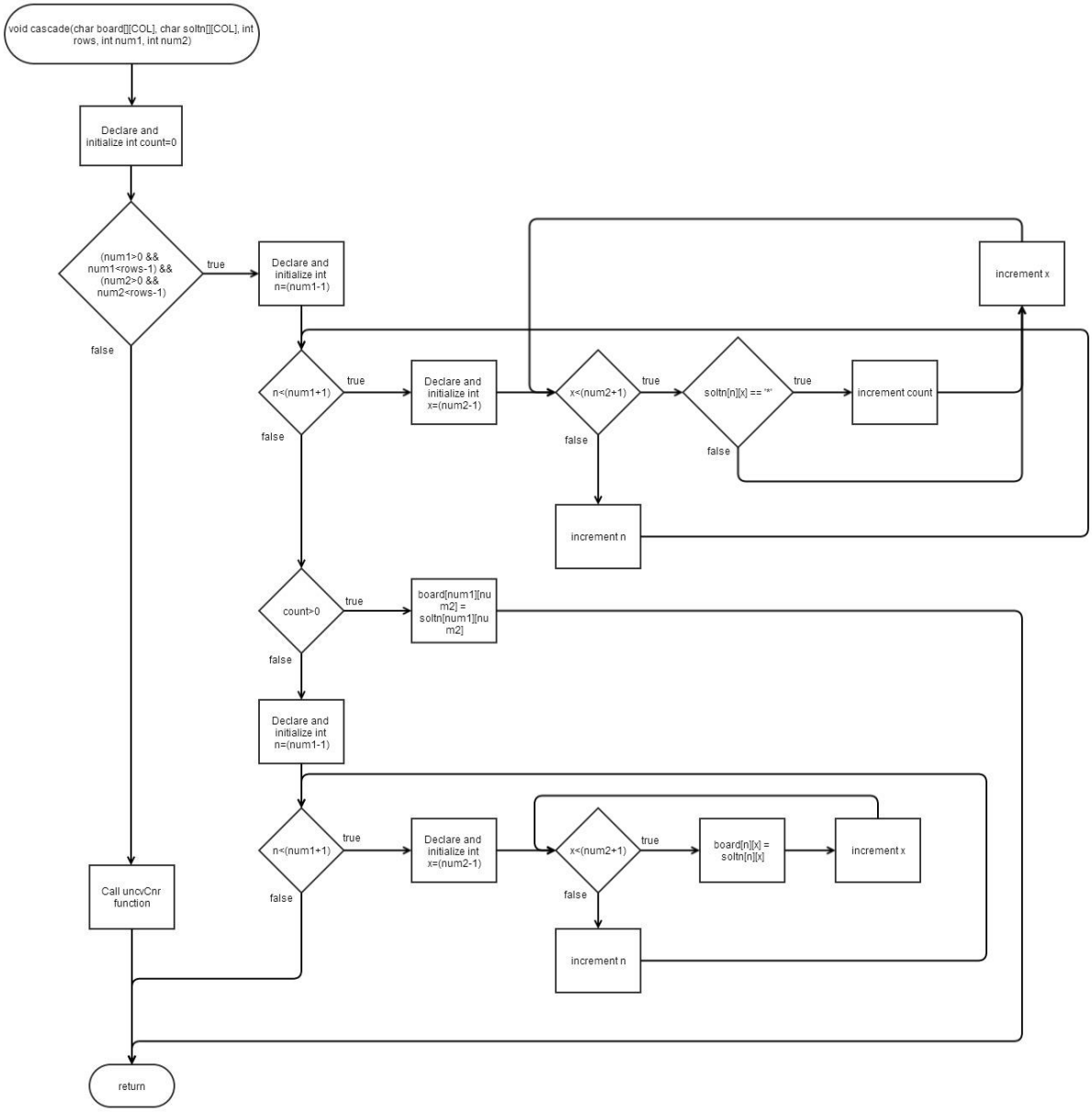

void chsLevel(int choice, int& mines, char soltn[][COL], int& rows, int& cols)
Declare ifstream game
choice == 1
true
Level chosen: Easy
open Game1.txt
set rows = cols = 9 mines = 10
call readSol function
false
choice == 2
true
Level chosen: Not so Easy
open Game2.txt
set rows = cols = 16 mines = 40
call readSol function
false
return

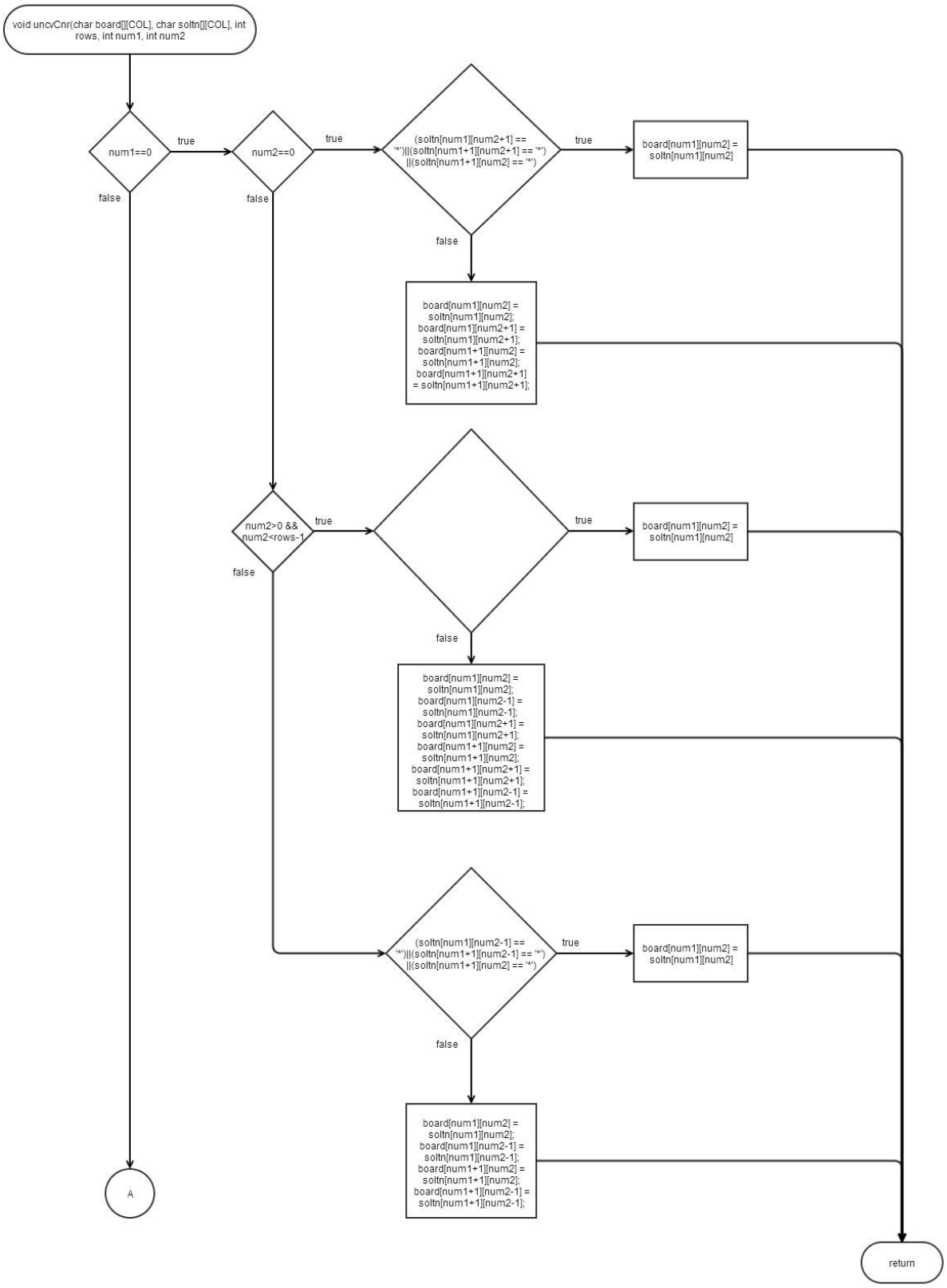

void play(char board[][COL], char soltn[][COL], int row, int col, int mines
Declare variables
char num1; int num2, num3; string winner;
ofstream allWnrs;
char *wnnrs=new char[row*col];
bool gameOvr,invInpt;
int begin =time(0);
Input which square to uncover, num1, num2
Call mkMove function, assign return value to invInpt
invInpt
true
false
Call winLose function, assign return value to gameOvr
Declare and initialize unsigned int count=0, int n=0
A
A
n<row
true
false
Declare and initialize int x=0
x<col
true
false
board[n][x] == soltn[n][x]
true
false
increment count
increment x
increment n
count == (row*col - mines)
true
false
Output won
Declare and initialize int end = time(0)
Call wnrsBrd function
Input winner
append a space to winner
open Winners.txt
Output winner to Winners.txt file
close Winners.txt
gameOvr = true;
!gameOvr
true
false
return

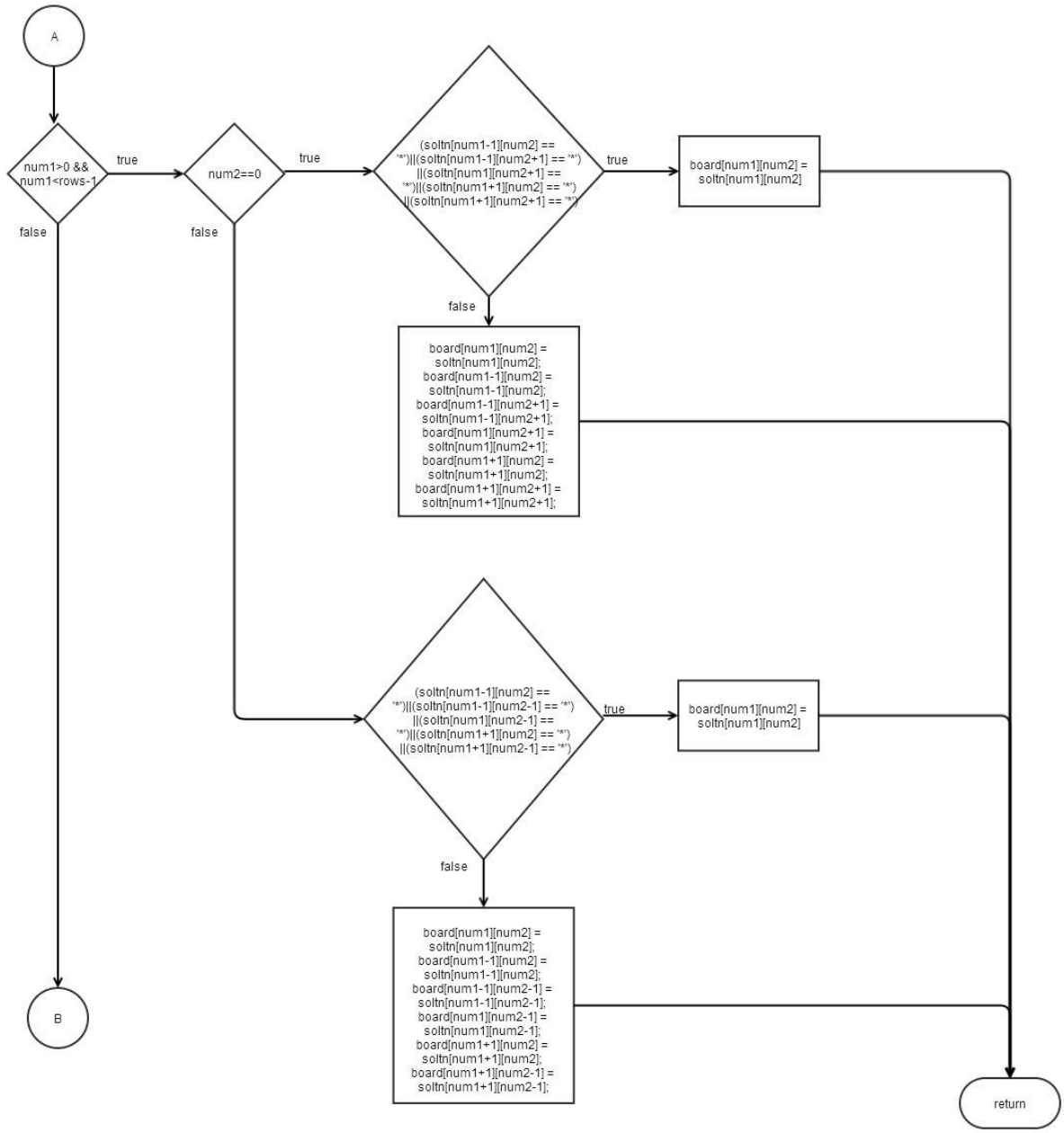

bool mkMove(char num1, int& num2, int& num3)
Force num1 to uppercase and assign it to num1
Subtract 'A' from num1 and static cast to int and assign to num3
Decrement num2
A
A
(num2<0||num2>cols-1)||(num3<0||num3>rows-1)
true
false
Invalid choice
return true
return false

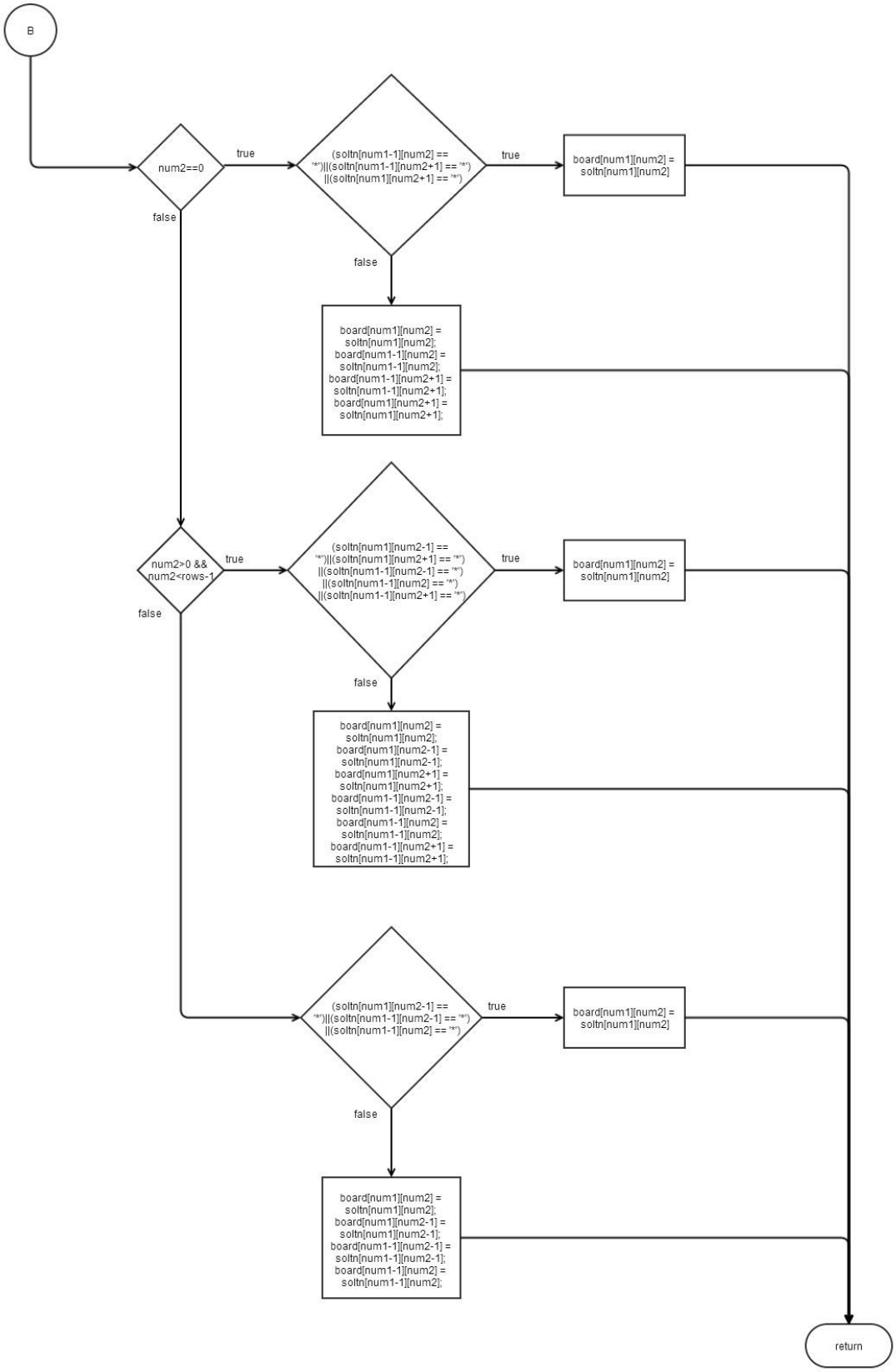

bool winLose(char board[][COL], char soltn[][COL], int rows, int cols, int num1, int num2)
soltn[num1][num2] == '*'
true
false
board[num1][num2] = soltn[num1][num2]
Call display function
Output lost and game over
return true
Call cascade function
Call display function
return false

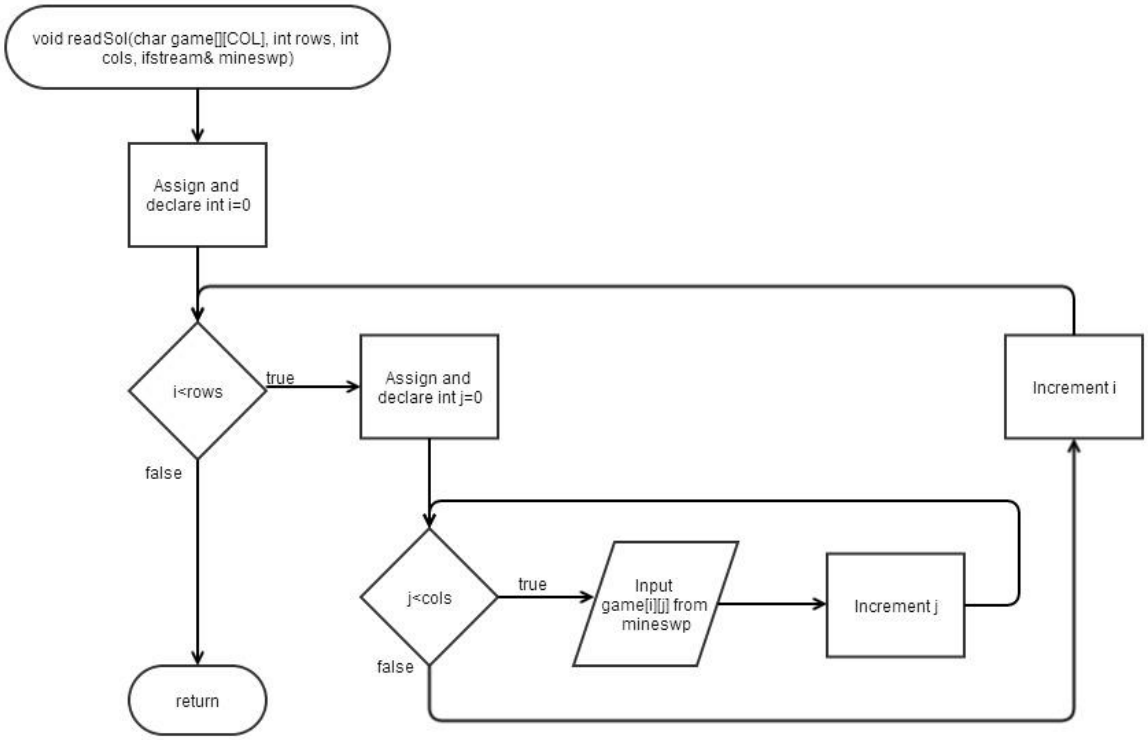

void display(char board[][COL], int rows, int cols)
Assign and declare char row = 'A'
Output space
Assign and declare int i=0
i<cols
true
false
setw(4) Output i+1
increment i
A
A
output endl endl
Assign and declare int *r=&rows, *c=&cols; int n=0
n<*r
Output row and 3 spaces
Assign and declare int x=0
x<*c
Output board[n][x] and 3 spaces
increment x
Output endl
increment row and n
return

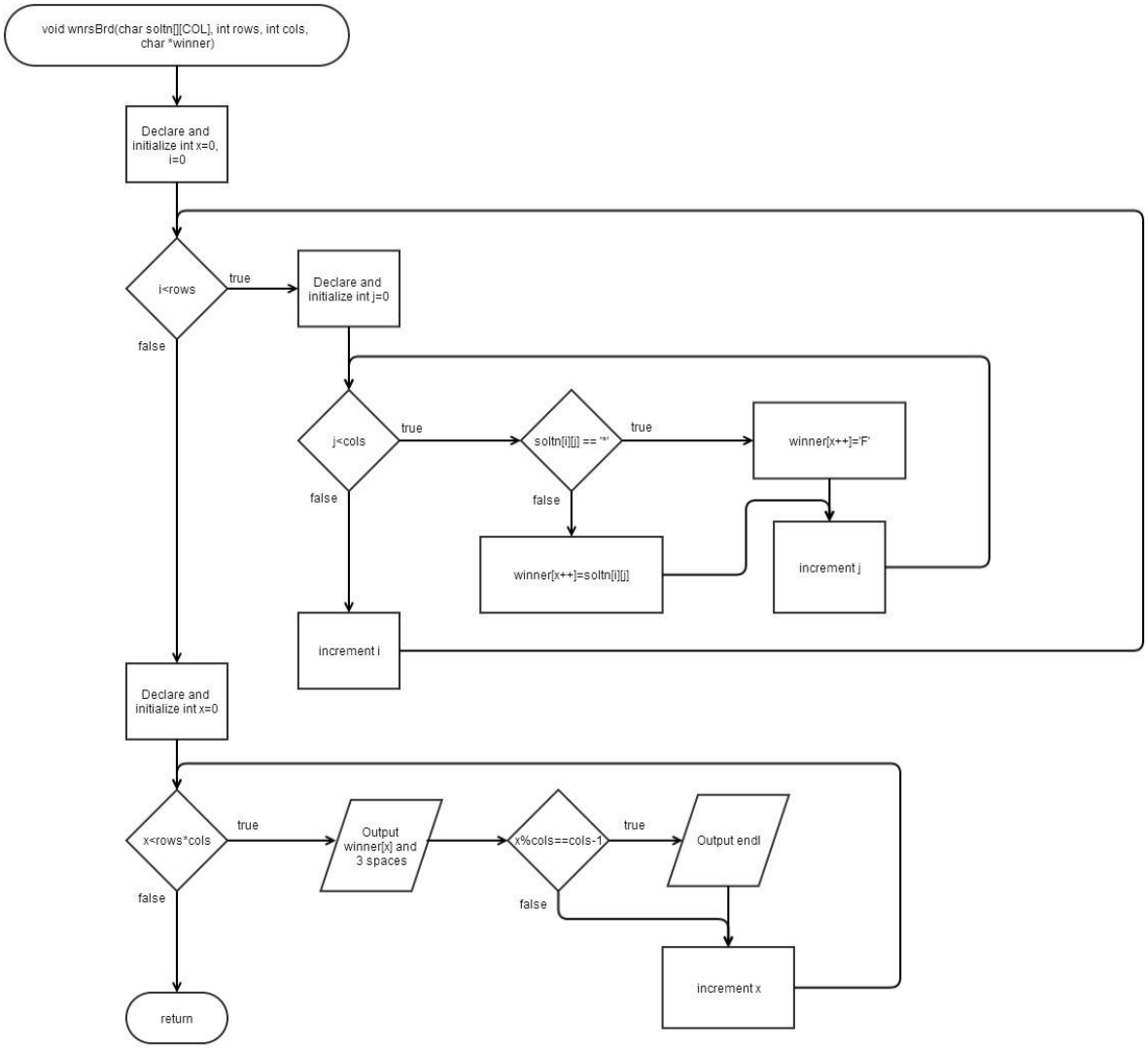

void cascade(char board[][COL], char soltn[][COL], int rows, int num1, int num2)
Declare and initialize int count=0
(num1>0 && num1<rows-1) && (num2>0 && num2<rows-1)
true
false
Declare and initialize int n=(num1-1)
n<(num1+1)
true
false
Declare and initialize int x=(num2-1)
x<(num2+1)
true
false
soltn[n][x] == '*'
true
false
increment count
increment x
increment n
count>0
true
false
board[num1][num2] = soltn[num1][num2]
Declare and initialize int n=(num1-1)
n<(num1+1)
true
false
Declare and initialize int x=(num2-1)
x<(num2+1)
true
false
board[n][x] = soltn[n][x]
increment x
increment n
Call uncvCnr function
return

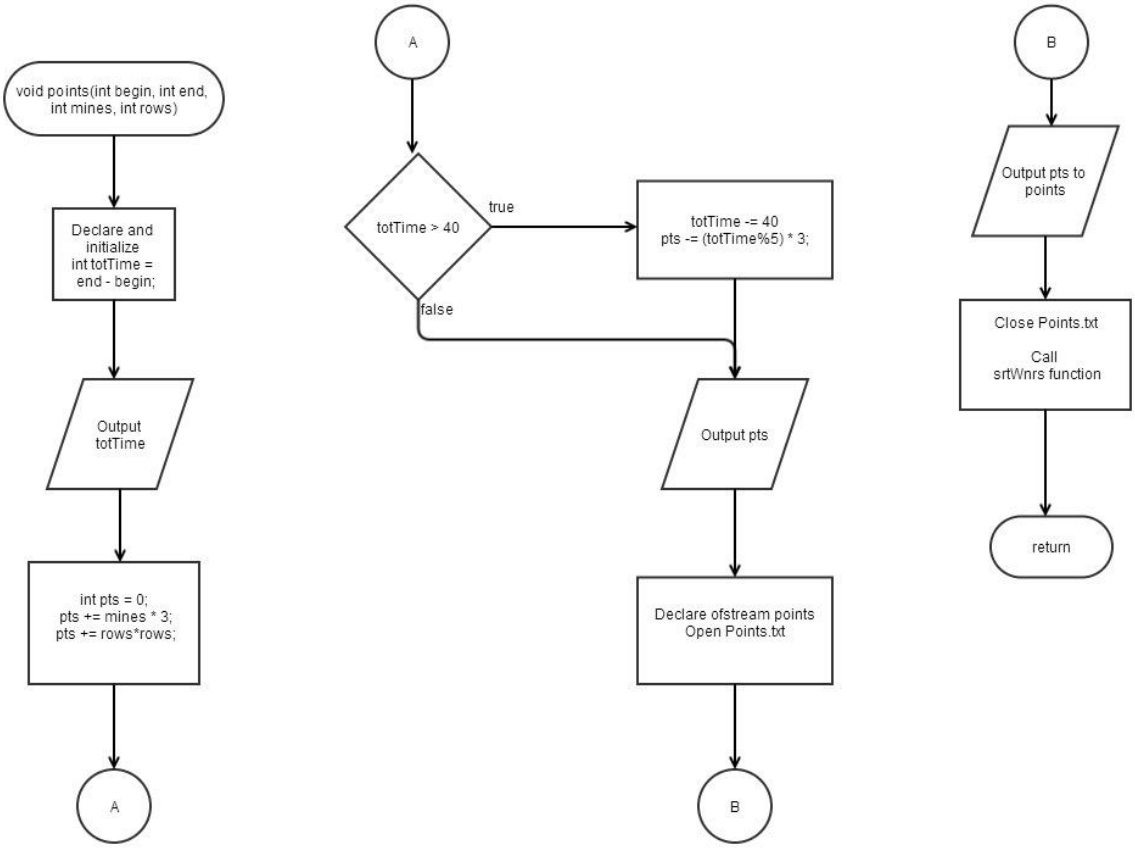

void uncvCnr(char board[][COL], char soltn[][COL], int rows, int num1, int num2
num1==0
true
false
num2==0
true
false
(soltn[num1][num2+1] == '*')||(soltn[num1+1][num2+1] == '*')||(soltn[num1+1][num2] == '*')
true
false
board[num1][num2] = soltn[num1][num2]
board[num1][num2] = soltn[num1][num2]; board[num1][num2+1] = soltn[num1][num2+1]; board[num1+1][num2] = soltn[num1+1][num2]; board[num1+1][num2+1] = soltn[num1+1][num2+1];
num2>0 && num2<rows-1
true
false
true
board[num1][num2] = soltn[num1][num2]
false
board[num1][num2] = soltn[num1][num2]; board[num1][num2-1] = soltn[num1][num2-1]; board[num1][num2+1] = soltn[num1][num2+1]; board[num1+1][num2] = soltn[num1+1][num2]; board[num1+1][num2+1] = soltn[num1+1][num2+1]; board[num1+1][num2-1] = soltn[num1+1][num2-1];
(soltn[num1][num2-1] == '*')||(soltn[num1+1][num2-1] == '*')||(soltn[num1+1][num2] == '*')
true
board[num1][num2] = soltn[num1][num2]
false
board[num1][num2] = soltn[num1][num2]; board[num1][num2-1] = soltn[num1][num2-1]; board[num1+1][num2] = soltn[num1+1][num2]; board[num1+1][num2-1] = soltn[num1+1][num2-1];
A
return

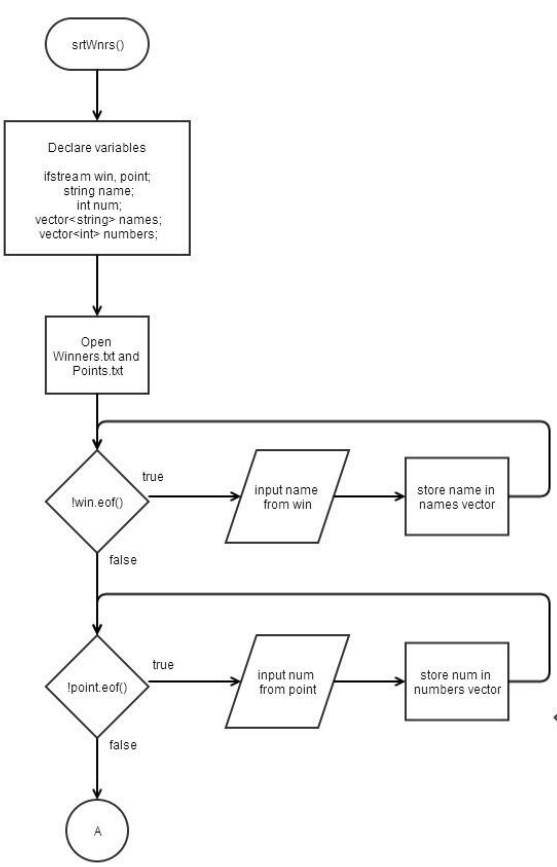

A
num1>0 && num1<rows-1
true
false
num2==0
true
false
(soltn[num1-1][num2] == '*')||(soltn[num1-1][num2+1] == '*') ||(soltn[num1][num2+1] == '*')||(soltn[num1+1][num2] == '*') ||(soltn[num1+1][num2+1] == '*')
true
board[num1][num2] = soltn[num1][num2]
false
board[num1][num2] = soltn[num1][num2]; board[num1-1][num2] = soltn[num1-1][num2]; board[num1-1][num2+1] = soltn[num1-1][num2+1]; board[num1][num2+1] = soltn[num1][num2+1]; board[num1+1][num2] = soltn[num1+1][num2]; board[num1+1][num2+1] = soltn[num1+1][num2+1];
(soltn[num1-1][num2] == '*')||(soltn[num1-1][num2-1] == '*') ||(soltn[num1][num2-1] == '*')||(soltn[num1+1][num2] == '*') ||(soltn[num1+1][num2-1] == '*')
true
board[num1][num2] = soltn[num1][num2]
false
board[num1][num2] = soltn[num1][num2]; board[num1-1][num2] = soltn[num1-1][num2]; board[num1-1][num2-1] = soltn[num1-1][num2-1]; board[num1][num2-1] = soltn[num1][num2-1]; board[num1+1][num2] = soltn[num1+1][num2]; board[num1+1][num2-1] = soltn[num1+1][num2-1];
B
return

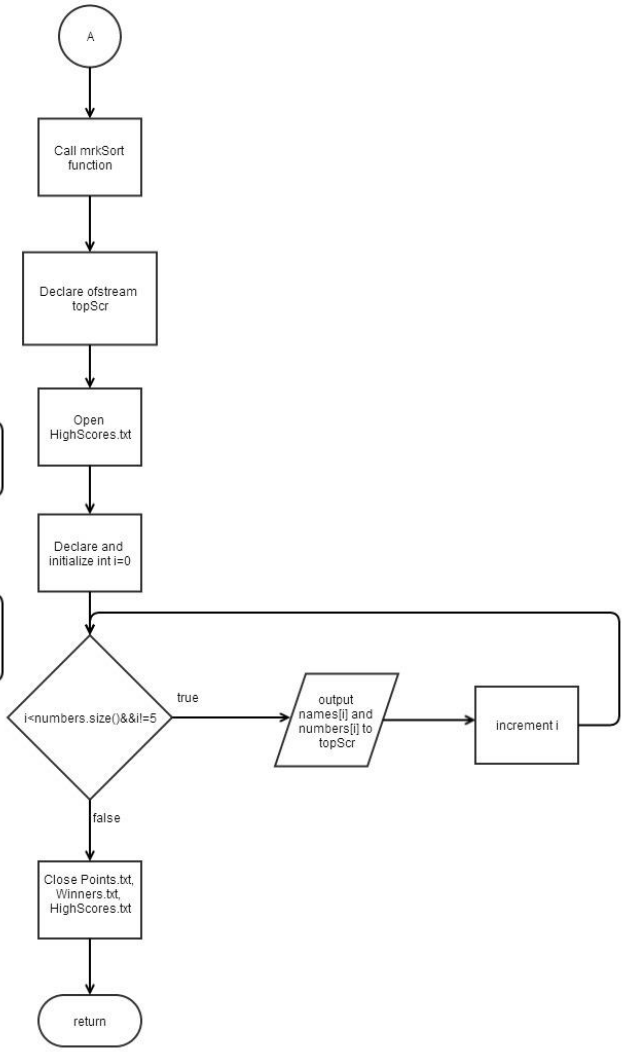

B
num2==0
true
false
(soltn[num1-1][num2] == '*')||(soltn[num1-1][num2+1] == '*') ||(soltn[num1][num2+1] == '*')
true
false
board[num1][num2] = soltn[num1][num2]
board[num1][num2] = soltn[num1][num2]; board[num1-1][num2] = soltn[num1-1][num2]; board[num1-1][num2+1] = soltn[num1-1][num2+1]; board[num1][num2+1] = soltn[num1][num2+1];
num2>0 && num2<rows-1
true
false
(soltn[num1][num2-1] == '*')||(soltn[num1][num2+1] == '*') ||(soltn[num1-1][num2-1] == '*') ||(soltn[num1-1][num2] == '*') ||(soltn[num1-1][num2+1] == '*')
true
false
board[num1][num2] = soltn[num1][num2]
board[num1][num2] = soltn[num1][num2]; board[num1][num2-1] = soltn[num1][num2-1]; board[num1][num2+1] = soltn[num1][num2+1]; board[num1-1][num2-1] = soltn[num1-1][num2-1]; board[num1-1][num2] = soltn[num1-1][num2]; board[num1-1][num2+1] = soltn[num1-1][num2+1];
(soltn[num1][num2-1] == '*')||(soltn[num1-1][num2-1] == '*') ||(soltn[num1-1][num2] == '*')
true
false
board[num1][num2] = soltn[num1][num2]
board[num1][num2] = soltn[num1][num2]; board[num1][num2-1] = soltn[num1][num2-1]; board[num1-1][num2-1] = soltn[num1-1][num2-1]; board[num1-1][num2] = soltn[num1-1][num2];
return

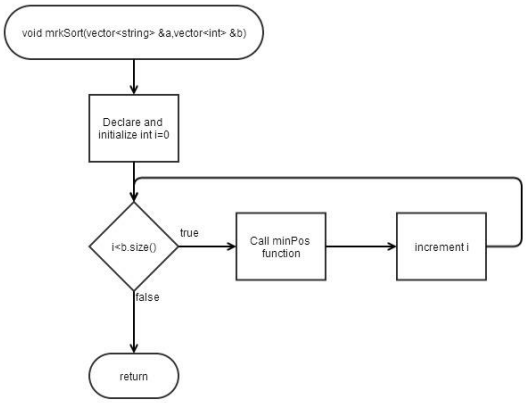

void readSol(char game[][COL], int rows, int cols, ifstream& mineswp)
Assign and declare int i=0
i<rows
true
false
Assign and declare int j=0
j<cols
true
false
Input game[i][j] from mineswp
Increment j
Increment i
return

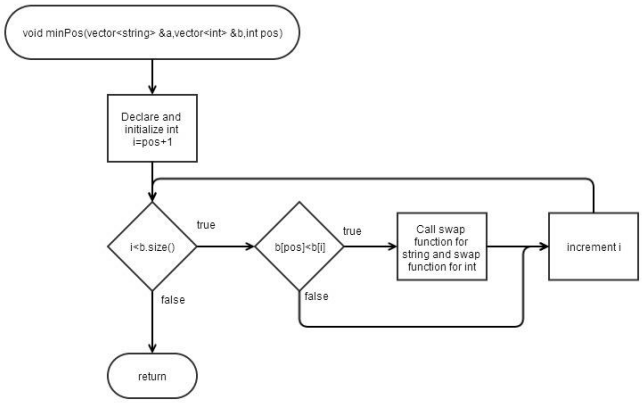

void wnrsBrd(char soltn[][COL], int rows, int cols, char *winner)
Declare and initialize int x=0, i=0
i<rows
true
false
Declare and initialize int j=0
j<cols
true
false
soltn[i][j] == '*'
true
false
winner[x++]='F'
winner[x++]=soltn[i][j]
increment j
increment i
Declare and initialize int x=0
x<rows*cols
true
false
Output winner[x] and 3 spaces
x%cols==cols-1
true
false
Output endl
increment x
return

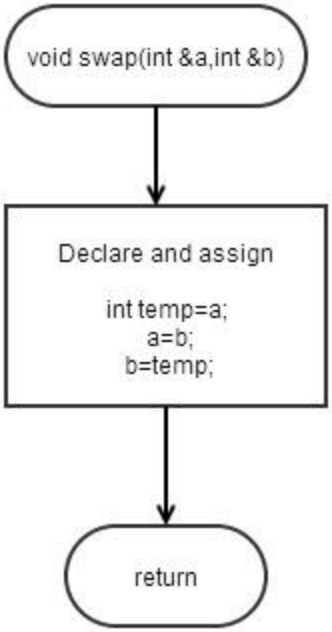

void points(int begin, int end, int mines, int rows)
Declare and initialize int totTime = end - begin;
Output totTime
int pts = 0; pts += mines * 3; pts += rows*rows;
A
A
totTime > 40
true
false
totTime -= 40 pts -= (totTime%5) * 3;
Output pts
Declare ofstream points Open Points.txt
B
B
Output pts to points
Close Points.txt
Call srtWnrs function
return

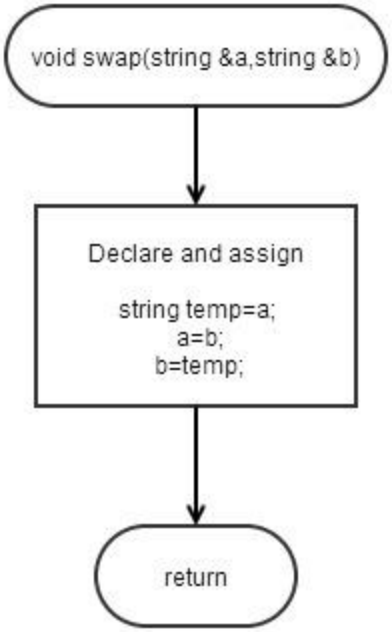

srtWnrs()
Declare variables
ifstream win, point;
string name;
int num;
vector<string> names;
vector<int> numbers;
Open Winners.txt and Points.txt
!win.eof()
true
input name from win
store name in names vector
false
!point.eof()
true
input num from point
store num in numbers vector
false
A


A
Call mrkSort function
Declare ofstream topScr
Open HighScores.txt
Declare and initialize int i=0
i<numbers.size()&&i!=5
true
output names[i] and numbers[i] to topScr
increment i
false
Close Points.txt, Winners.txt, HighScores.txt
return


void mrkSort(vector<string> &a,vector<int> &b)
Declare and initialize int i=0
i<b.size()
true
Call minPos function
increment i
false
return


void minPos(vector<string> &a,vector<int> &b,int pos)
Declare and initialize int i=pos+1
i<b.size()
true
b[pos]<b[i]
true
Call swap function for string and swap function for int
increment i
false
false
return




# Sample Input/Output

Output

Minesweeper_Project1_CSC5_RCC_013014 (Build) | Minesweeper_Project

```
Choose the difficulty by typing in the corresponding nu
   1) Easy
   2) Not so Easy
1
You have chosen Level: Easy

    1   2   3   4   5   6   7   8   9

A   M   M   M   M   M   M   M   M   M
B   M   M   M   M   M   M   M   M   M
C   M   M   M   M   M   M   M   M   M
D   M   M   M   M   M   M   M   M   M
E   M   M   M   M   M   M   M   M   M
F   M   M   M   M   M   M   M   M   M
G   M   M   M   M   M   M   M   M   M
H   M   M   M   M   M   M   M   M   M
I   M   M   M   M   M   M   M   M   M

Choose which square to uncover (i.e. A1) i9

    1   2   3   4   5   6   7   8   9

A   M   M   M   M   M   M   M   M   M
B   M   M   M   M   M   M   M   M   M
C   M   M   M   M   M   M   M   M   M
D   M   M   M   M   M   M   M   M   M
E   M   M   M   M   M   M   M   M   M
F   M   M   M   M   M   M   M   M   M
G   M   M   M   M   M   M   M   M   M
H   M   M   M   M   M   M   M   1   0
I   M   M   M   M   M   M   M   1   0

Choose which square to uncover (i.e. A1)
```

Output

Minesweeper_Project1_CSC5_RCC_013014 (Build) | Minesweeper_Pr

```
Choose which square to uncover (i.e. A1) f5

    1   2   3   4   5   6   7   8   9

A   M   M   M   M   M   M   M   M   M
B   M   M   M   M   M   M   M   M   M
C   1   1   M   M   M   M   M   M   M
D   0   0   M   M   M   M   M   M   M
E   0   0   M   2   3   3   M   M   M
F   M   M   M   1   0   1   M   M   M
G   M   M   M   1   0   1   M   M   M
H   0   1   M   1   0   1   M   1   0
I   0   0   M   0   0   1   M   1   0

Choose which square to uncover (i.e. A1) d5

    1   2   3   4   5   6   7   8   9

A   M   M   M   M   M   M   M   M   M
B   M   M   M   M   M   M   M   M   M
C   1   1   M   M   M   M   M   M   M
D   0   0   M   M   *   M   M   M   M
E   0   0   M   2   3   3   M   M   M
F   M   M   M   1   0   1   M   M   M
G   M   M   M   1   0   1   M   M   M
H   0   1   M   1   0   1   M   1   0
I   0   0   M   0   0   1   M   1   0

Game Over. You lose!

RUN SUCCESSFUL (total time: 2m 54s)
```

Output

Minesweeper_Project1_CSC5_RCC_013014 (Build) | Minesweeper_Project1_CS

```
    1   2   3   4   5   6   7   8   9

A   1   1   1   1   M   2   1   1   0
B   1   M   M   M   M   M   M   1   0
C   1   M   M   M   M   M   M   M   M
D   0   0   M   M   M   M   M   M   M
E   0   0   M   M   M   M   M   M   M
F   0   1   M   M   M   M   M   M   M
G   0   1   M   M   M   M   M   M   1
H   0   1   M   M   M   M   M   1   0
I   0   0   M   M   M   M   M   1   0

Choose which square to uncover (i.e. A1) f9

    1   2   3   4   5   6   7   8   9

A   1   1   1   1   M   2   1   1   0
B   1   M   M   M   M   M   M   1   0
C   1   M   M   M   M   M   M   M   M
D   0   0   M   M   M   M   M   M   M
E   0   0   M   M   M   M   M   M   M
F   0   1   M   M   M   M   M   M   1
G   0   1   M   M   M   M   M   M   1
H   0   1   M   M   M   M   M   1   0
I   0   0   M   M   M   M   M   1   0

Choose which square to uncover (i.e. A1) i10
    Invalid choice. Please try again.

Choose which square to uncover (i.e. A1) j5
    Invalid choice. Please try again.

Choose which square to uncover (i.e. A1)
```

Output

Minesweeper_Project1_CSC5_RCC_013014 (Build) | Minesweeper_Project1_CSC5_RC

```
I   0   0   0   0   0   1   M   1   0

Choose which square to uncover (i.e. A1) g8

    1   2   3   4   5   6   7   8   9

A   1   1   1   1   M   2   1   1   0
B   1   M   1   1   1   2   M   1   0
C   1   1   2   2   3   2   2   1   0
D   0   0   1   M   M   M   1   0   0
E   0   0   1   2   3   3   2   2   1
F   0   1   1   1   0   1   M   M   1
G   0   1   M   1   0   1   2   2   1
H   0   1   1   1   0   1   1   1   0
I   0   0   0   0   0   1   M   1   0

Congratulations! You win!

1   1   1   1   F   2   1   1   0
1   F   1   1   1   2   F   1   0
1   1   2   2   3   2   2   1   0
0   0   1   F   F   F   1   0   0
0   0   1   2   3   3   2   2   1
0   1   1   1   0   1   F   F   1
0   1   F   1   0   1   2   2   1
0   1   1   1   0   1   1   1   0
0   0   0   0   0   1   F   1   0
Please enter your name to record you as a winner: Victoria
You completed this level in 148 seconds.
You have been awarded 102 points!

RUN SUCCESSFUL (total time: 3m 25s)
```

# Data Types Used

| Data Type | Example | Description | Location |
|---|---|---|---|
| **int** | mines | Number of mines on board, depending on difficulty | main() |
| **char** | board[][COLS] | Array for game board, able to hold (in the form of a char) numbers, letters and asterisks | cascade() |
| **bool** | invInpt | Assigned true if values for selecting tile are not in the correct range | play() |
| **ifstream** | game | Used to open appropriate game level from text file | chsLevel() |
| **ofstream** | points | Used to write awarded points to a file | points() |
| **string** | vector<>names | Store names of winners | srtWnrs() |
| **const unsigned int** | rows | Maximum rows | main() |
| **unsigned int** | count | Keep track of squares uncovered | play() |

# C++ Constructs (Savitch, 8th Edition)

| Chapter | Construct | Location |
|---|---|---|
| **2** | Variables | main |
| | Data types | main |
| | Input/output | main |
| | Formatting/iomanip | display |
| | string | play |
| | if, if-else, if-else-if | uncvrCnr |
| | do-while | play |
| | increment | cascade |
| | decrement | mkMove |
| **3** | Boolean expression | cascade |
| | switch | chsLevel |
| | For loop | initial |
| **4** | Type casting | mkMove |
| | Functions, function calls | main |
| | Call by value | winLose |
| | Global constants | global |
| **5** | Void functions | initial |
| | Call by reference | chsLevel |
| | Functions calling functions | play |
| **6** | File i/o | chsLevel, play |
| | Appending file | play |
| | Streams as arguments | readSol |
| | toupper | mkMove |
| **7** | Arrays, multidimensional | cascade |
| **8** | vectors | wnrsBrd |
| **9** | Dynamic arrays | play() |
| | Pointers | display() |

# References

I used the following site to get started by understanding the rules for which tiles are uncovered:

- http://ajithsimha.wordpress.com/2010/11/16/how-does-minesweeper-work/
- Savitch textbook, 8th Edition
- Mark_Sort program

# Source Code

```
/*
 * File:   main.cpp
 * Author: Victoria Hodnett
 * Created on January 30, 2014, 11:23 AM
 * WIN14 CSC5 RCC Project 1
 * Minesweeper
 */

//System Libraries
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <cstdlib>
#include <ctime>
using namespace std;

//Global Constants
const unsigned int COL = 26;

//Function Prototypes
void initial(char[][COL],int,int);
void chsLevel(int,int&,char[][COL],int&,int&);
void play(char[][COL],char[][COL],int,int,int);
bool mkMove(char,int&,int&,int,int);
bool winLose(char[][COL], char[][COL], int,int, int, int);
void display(char[][COL],int,int);
void cascade(char[][COL],char[][COL],int,int,int);
void uncvCnr(char[][COL],char[][COL],int,int,int);
void readSol(char[][COL],int,int,ifstream&);
void wnrsBrd(char[][COL],int,int,char*);
void points(int,int,int,int);
void srtWnrs();
void swap(int &,int &);
void swap(string &,string &);
void minPos(vector<string> &,vector<int> &,int);
void mrkSort(vector<string> &,vector<int> &);
```

```
//Execution Begins
int main(int argc, char** argv) {
    //Declare variables
    const unsigned int rows = 26;
    int choice,mines,actlRow,actlCol;
    char board[rows][COL], soltn[rows][COL];
    //Start game
    cout << "Welcome to Minesweeper!" << endl;
    //Choose level
    cout << "Choose the difficulty by typing in the corresponding "
            "number:" << endl;
    cout << "   1) Easy" << endl;
    cout << "   2) Not so Easy" << endl;
    cin >> choice;
    chsLevel(choice,mines,soltn,actlRow,actlCol);
    //Initialize board
    initial(board,actlRow,actlCol);
    //Display board
    display(board,actlRow,actlCol);
    //Start play
    play(board,soltn,actlRow,actlCol,mines);
    //Exit
    return 0;
}

//Initialize game board
//Input
//      board, row, col
//Output
//      none
//      reference board array
void initial(char board[][COL],int row,int col){
    for(int n=0; n<row; n++){
        for(int x=0; x<col; x++){
            board[n][x] = 'M';
        }
    }
}

//Begin play
//Input
//      board, soltn, row, col, mines
//Output
//      none
//      reference board array
void play(char board[][COL],char soltn[][COL],int row,int col,int mines){
    char num1;
    int num2,num3;
    string winner;
    ofstream allWnrs;
    char *wnnrs = new char[row*col];
    bool gameOvr,invInpt;
    int begin = time(0);
    do{
        do{
            //Make move
            cout << "Choose which square to uncover (i.e. A1) ";
            cin >> num1 >> num2;
```

```
            invInpt = mkMove(num1,num2,num3,row,col);
            cout << endl;
        }while(invInpt);
        //Determine if lost or may continue
        gameOvr = winLose(board,soltn,row,col,num3,num2);
        //Determine if won
        unsigned int count = 0;
        for (int n=0; n<row; n++){
            for(int x=0; x<col; x++){
                if(board[n][x] == soltn[n][x]) count++;
            }
        }
        if(count == ((row*col)-mines)){
            cout << "Congratulations! You win!" << endl;
            cout << endl;
            int end = time(0);
            //Print out winners board with flags
            wnrsBrd(soltn,row,col,wnnrs);
            cout << "Please enter your name to record you as a winner: ";
            cin >> winner;
            winner += " ";
            allWnrs.open("Winners.txt",ios::app);
            allWnrs << winner << endl;
            allWnrs.close();
            //Display time and score
            points(begin,end,mines,row);
            gameOvr = true;
        }
    }while(!gameOvr);
}

//Manipulate choice user made to
//determine move
//Validate choice
//Input
//      num1, num2, num3,rows,cols
//Output
//      bool = true or false
//      reference num2,num3
bool mkMove(char num1, int& num2,int& num3,int rows,int cols){
    num1 = toupper(num1);
    num3 = static_cast<int>(num1 - 'A');
    num2--;
    if ((num2<0||num2>cols-1)||(num3<0||num3>rows-1)){
        cout << "    Invalid choice. Please try again." << endl;
        return true;
    }
    else
        return false;
}

//Determine if won or may continue
//Input
//      board, soltn, rows, cols, num1, num2
//Output
//      none
//      reference board
bool winLose(char board[][COL], char soltn[][COL], int rows,
```

```
                                int cols, int num1, int num2){
    if (soltn[num1][num2] == '*'){
        board[num1][num2] = soltn[num1][num2];
        display(board, rows,cols);
        cout << "Game Over. You lose!" << endl;
        return true;
    }else{
        cascade(board,soltn,rows,num1,num2);
        display(board,rows,cols);
        return false;
    }

}

//Display game board
//Input
//      board, rows, cols
//Output
//      none
void display(char board[][COL], int rows, int cols){
    //Print out A-I for rows, 1-9 for columns
    char row = 'A';
    cout << " ";
    for(int i=0; i<cols; i++){
        cout << setw(4);
        cout << i+1;
    }
    cout << endl << endl;
    int *r=&rows,*c=&cols;
    for(int n=0; n<*r; n++){
        cout << row << "   ";
        for(int x=0; x<*c; x++){
            cout << board[n][x] << "   ";
        }
        cout << endl;
        row++;
    }
    cout << endl;
}

//Uncover tiles, determine if more than one
//may be uncovered
//Input
//      board, soltn, rows, num1, num2
//Output
//      none
//      reference board
void cascade(char board[][COL], char soltn[][COL],
                     int rows, int num1, int num2){
    int count = 0;
    //Test center of board
    //If any mines surround the chosen tile, only
    //uncover the chosen tile
    if ((num1>0 && num1<rows-1)&&(num2>0 && num2<rows-1)){
        for(int n=(num1-1); n<=(num1+1); n++){
            for(int x=(num2-1); x<=(num2+1); x++){
                if (soltn[n][x] == '*'){
                    count++;
```

```
                }
            }
        }
        if(count > 0){
            board[num1][num2] = soltn[num1][num2];
            return;
        }
        //If no mines surround the chosen tile, uncover
        //all adjacent to the chosen tile
        else{
            for(int n=(num1-1); n<=(num1+1); n++){
                for(int x=(num2-1); x<=(num2+1); x++){
                    board[n][x] = soltn[n][x];
                }
            }
        }
    //Test corners, only a partial cascade to remain in bounds of array
    }else{
        uncvCnr(board,soltn,rows,num1,num2);
    }
}

//Test corners, uncover only what's in the array
//Input
//      board, soltn, rows, num1, num2
//Output
//      none
//      reference board
void uncvCnr(char board[][COL], char soltn[][COL],
                     int rows, int num1, int num2){
    if(num1==0){
            if(num2==0){
                if((soltn[num1][num2+1] == '*')||(soltn[num1+1][num2+1] == '*')
                    ||(soltn[num1+1][num2] == '*')){
                    board[num1][num2] = soltn[num1][num2];
                    return;
                }else{
                    board[num1][num2] = soltn[num1][num2];
                    board[num1][num2+1] = soltn[num1][num2+1];
                    board[num1+1][num2] = soltn[num1+1][num2];
                    board[num1+1][num2+1] = soltn[num1+1][num2+1];
                }
            }else if(num2>0 && num2<rows-1){
                if((soltn[num1][num2-1] == '*')||(soltn[num1][num2+1] == '*')
                    ||(soltn[num1+1][num2-1] == '*')
                    ||(soltn[num1+1][num2+1] == '*')
                    ||(soltn[num1+1][num2] == '*')){
                    board[num1][num2] = soltn[num1][num2];
                    return;
                }else{
                    board[num1][num2] = soltn[num1][num2];
                    board[num1][num2-1] = soltn[num1][num2-1];
                    board[num1][num2+1] = soltn[num1][num2+1];
                    board[num1+1][num2] = soltn[num1+1][num2];
                    board[num1+1][num2+1] = soltn[num1+1][num2+1];
                    board[num1+1][num2-1] = soltn[num1+1][num2-1];
                }
            }else{//if num2==8
```

```
            if((soltn[num1][num2-1] == '*')||(soltn[num1+1][num2-1] == '*')
                ||(soltn[num1+1][num2] == '*')){
                board[num1][num2] = soltn[num1][num2];
                return;
            }else{
                board[num1][num2] = soltn[num1][num2];
                board[num1][num2-1] = soltn[num1][num2-1];
                board[num1+1][num2] = soltn[num1+1][num2];
                board[num1+1][num2-1] = soltn[num1+1][num2-1];
            }
        }
    }else if(num1>0 && num1<rows-1){
        if(num2==0){
            if((soltn[num1-1][num2] == '*')||(soltn[num1-1][num2+1] == '*')
                ||(soltn[num1][num2+1] == '*')||(soltn[num1+1][num2] == '*')
                ||(soltn[num1+1][num2+1] == '*')){
                board[num1][num2] = soltn[num1][num2];
                return;
            }else{
                board[num1][num2] = soltn[num1][num2];
                board[num1-1][num2] = soltn[num1-1][num2];
                board[num1-1][num2+1] = soltn[num1-1][num2+1];
                board[num1][num2+1] = soltn[num1][num2+1];
                board[num1+1][num2] = soltn[num1+1][num2];
                board[num1+1][num2+1] = soltn[num1+1][num2+1];
            }
        }else{//if num2==8
            if((soltn[num1-1][num2] == '*')||(soltn[num1-1][num2-1] == '*')
                ||(soltn[num1][num2-1] == '*')||(soltn[num1+1][num2] == '*')
                ||(soltn[num1+1][num2-1] == '*')){
                board[num1][num2] = soltn[num1][num2];
                return;
            }else{
                board[num1][num2] = soltn[num1][num2];
                board[num1-1][num2] = soltn[num1-1][num2];
                board[num1-1][num2-1] = soltn[num1-1][num2-1];
                board[num1][num2-1] = soltn[num1][num2-1];
                board[num1+1][num2] = soltn[num1+1][num2];
                board[num1+1][num2-1] = soltn[num1+1][num2-1];
            }
        }
    }else{//if num1==8
        if(num2==0){
            if((soltn[num1-1][num2] == '*')||(soltn[num1-1][num2+1] == '*')
                ||(soltn[num1][num2+1] == '*')){
                board[num1][num2] = soltn[num1][num2];
                return;
            }
            else{
                board[num1][num2] = soltn[num1][num2];
                board[num1-1][num2] = soltn[num1-1][num2];
                board[num1-1][num2+1] = soltn[num1-1][num2+1];
                board[num1][num2+1] = soltn[num1][num2+1];
            }
        }else if(num2>0 && num2<rows-1){
            if((soltn[num1][num2-1] == '*')||(soltn[num1][num2+1] == '*')
                ||(soltn[num1-1][num2-1] == '*')
                ||(soltn[num1-1][num2] == '*')
```

```
                    ||(soltn[num1-1][num2+1] == '*')){
                    board[num1][num2] = soltn[num1][num2];
                    return;
                }else{
                    board[num1][num2] = soltn[num1][num2];
                    board[num1][num2-1] = soltn[num1][num2-1];
                    board[num1][num2+1] = soltn[num1][num2+1];
                    board[num1-1][num2-1] = soltn[num1-1][num2-1];
                    board[num1-1][num2] = soltn[num1-1][num2];
                    board[num1-1][num2+1] = soltn[num1-1][num2+1];
                }
            }else{//if num2==8
                if((soltn[num1][num2-1] == '*')||(soltn[num1-1][num2-1] == '*')
                    ||(soltn[num1-1][num2] == '*')){
                    board[num1][num2] = soltn[num1][num2];
                    return;
                }else{
                    board[num1][num2] = soltn[num1][num2];
                    board[num1][num2-1] = soltn[num1][num2-1];
                    board[num1-1][num2-1] = soltn[num1-1][num2-1];
                    board[num1-1][num2] = soltn[num1-1][num2];
                }
            }
        }
}

//Choose level and assign board accordingly
//Input
//      choice, mines, soltn, rows, cols
//Output
//      none
//      reference mines, rows, cols
void chsLevel(int choice, int& mines, char soltn[][COL], int& rows, int& cols){
    ifstream game;
    switch(choice){
        case 1:{
            cout << "You have chosen Level: Easy " << endl;
            cout << endl;
            game.open("Game1.txt");
            rows = cols = 9;
            mines = 10;
            //Read in solution
            //read from file into soltn array
            readSol(soltn,rows,cols,game);
            break;}
        case 2:{
            cout << "You have chosen Level: Not so Easy " << endl;
            cout << endl;
            game.open("Game2.txt");
            rows = cols = 16;
            mines = 40;
            readSol(soltn,rows,cols,game);
            break;}
        default:
            break;
    }
}
```

```
//Read in solution from file
//Input
//      game, rows, cols, mineswp
//Output
//      none
//      reference game
void readSol(char game[][COL], int rows, int cols, ifstream& mineswp){
    for(int i=0; i<rows; i++){
        for(int j=0; j<cols; j++){
            mineswp >> game[i][j];
        }
    }
    mineswp.close();
    //Test input
//    for(int i=0; i<ROWS; i++){
//        for(int j=0; j<COL; j++){
//            cout << game[i][j];
//        }
//        cout << endl;
//    }
}

//Display flags on the winners board
//Input
//      soltn, rows, cols, winner
//Output
//      none
//      reference winner
void wnrsBrd(char soltn[][COL],int rows,int cols,char *winner){
    int x=0;
    for(int i=0; i<rows; i++){
        for(int j=0; j<cols;j++){
            if(soltn[i][j]=='*')winner[x++]='F';
            else winner[x++]=soltn[i][j];
        }
    }
    //Output winners board
    for(int x=0; x<rows*cols; x++){
        cout << winner[x] << "   ";
        if(x%cols==cols-1){
            cout << endl;
        }
    }
}

//Calculate total time and points awarded
//Display points
//Input
//      begin, end, mines, rows
//Output
//      none
void points(int begin, int end, int mines, int rows){
    //Display amount of time
    int totTime =  end - begin;
    cout << "You completed this level in "
            << totTime << " seconds. " << endl;
    //Calculate points
    int pts = 0;
```

```
    //2 points for every mine survived
    pts += mines * 3;
    //1 point for each tile on the board
    pts += rows*rows;
    //Subtract points for time
    if (totTime > 40){
        totTime -= 40;
        //Subtract 3 points for every 5 seconds over 40
        pts -= (totTime%5) * 3;
    }
    //Display points
    cout << "You have been awarded " << pts << " points!" << endl;
    //Write points to file
    ofstream points;
    points.open("Points.txt",ios::app);
    points << setw(4);
    points << pts;
    points.close();
    //Sort winners
    srtWnrs();
}

//Sort winners and print top 5 high scores
//to file
//Input
//      none
//Output
//      none
void srtWnrs(){
    ifstream win, point;
    string name;
    int num;
    vector<string> names;
    vector<int> numbers;
    win.open("Winners.txt");
    point.open("Points.txt");
    //Read in values into vectors
    while(!win.eof()){
        win >> name;
        names.push_back(name);
    }

    while(!point.eof()){
        point >> num;
        numbers.push_back(num);
    }
    //Sort points in order from highest to lowest
    mrkSort(names,numbers);

//    //Print out all scores to console
//    cout << endl;
//    cout << "Order of Scores" << endl << endl;
//    for(int i=0; i<numbers.size(); i++){
//        cout << names[i] << " ";
//        cout << numbers[i];
//        cout << endl;
//    }
    //Write top 5 scores to file
```

```
    ofstream topScr;
    topScr.open("HighScores.txt");
    for(int i=0;(i<numbers.size()&&i!=5);i++){
        topScr << i+1 << ". " << names[i]
                << ": " << numbers[i] << " " << endl;
    }
    win.close();
    point.close();
    topScr.close();
}

//Sort scores and names from highest score
//to lowest score
//For functions
//      mrkSort,minPos,swap,swap
void mrkSort(vector<string> &a,vector<int> &b){
    for(int i=0;i<b.size()-1;i++){
        minPos(a,b,i);
    }
}

void minPos(vector<string> &a,vector<int> &b,int pos){
    for(int i=pos+1;i<b.size();i++){
        if(b[pos]<b[i]){
            swap(a[pos],a[i]);
            swap(b[pos],b[i]);
        }
    }
}

void swap(int &a,int &b){
    int temp=a;
    a=b;
    b=temp;
}

void swap(string &a,string &b){
    string temp=a;
    a=b;
    b=temp;
}
```