جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

CS207 Programming Methodology and Abstractions

Fall 2020: Midterm

Instructions:

1. The total points of this midterm are 100. The points of each exercise appear on the corresponding title.
2. Submission deadline is 22/10/2020 at 2:45 pm.
3. Use the best of your skills in code structure and naming conventions.
4. Submit only source code (no executable, no screen shots). Each program file should be named after the exercise it solves. Wrap all your files in one folder and rename it as follows: yourname_midterm, then zip it with the same name and submit it to the blackboard website of the course.

Exercise 1 (50 pts):

Recall the "tokens" exercise from the second assignment, where you wrote a function which prints to the screen the tokens of its string parameter, each on a separate line.

Modify the function such that it, instead, (or besides) places these tokens in a dynamically-allocated array of strings and provides them back to the calling function via the parameters.

The function protoype should be:

```
void intoTokens(string str, string* &tokens, int &nbrTokens)
```

Modify the main function as well with the proper function call.

Exercise 2 (50 pts):

Attached is a code scaffold for a console version of the minesweeper game http://minesweeperonline.com/ . The game consists in a 2D grid of cells, called also mine field, which content is initially hidden to the player (cell content is invisible). Each cell could contain either a mine or the number of mines among its 8 neighboring cells. The purpose of the game is to find out where the mines are located. The player can click (step) on a cell to reveal its content. If the cell contains a mine the game is over. If the cell does not contain a mine, the cell reveals the number of mines among its 8 neighboring cells. Based on the numbers in the revealed cells, the player can start guessing the mine locations. If the player is sure of the location of a mine, he can mark it as mine. The player wins once he reveals all non-mine cells.

The attached code is missing a snippet in the main function used to generate the initial state of the mine field. The objective is to write this snippet which consists in:

- Prompting the user and getting the width and height of the mine filed. If the user enters numbers larger than the maximum height or width predefined in the code. These maxima are used instead.
- Dynamically allocating a 2D array of cells
- Calculating the number of mines in the mine field
- Randomly placing the mines in the mine field. The function rand() from the library <cstdlib> generates a random integer in the range [0 .. MAXRAND] where MAXRAND is system-dependent. You can use this function to generate the random positions of the mines within the minefield.
- Filling the rest of the grid with the information about the number of mines in the 8 neighboring cells.

Each cell of the grid needs to hold more than one piece of information:

- Its content whether it is a mine or the number of neighboring cell: this is an integer which value should be -1 if the cell contains a mine and a number between 0 and 8 if it does not contain a mine
- A Boolean that tells whether the cell content is still hidden from the player or it has been revealed: this is mainly used by the `displayMineField` function to decide whether to print an empty square or to print it with its content. When the content is a mine (-1), the display function prints a star (*). At the beginning of the game, all the cells are unrevealed/hidden.
- A Boolean that tells whether the player had marked this cell as a mine: this is mainly used to prevent the player from revealing a cell that he already marked as a mine but also marked cells will be displayed as a square containing the letter 'M'. At the beginning of the game, all the cells are unmarked.

Thus, the struct

```
struct cell {
      int value;
      bool visible; // whether the player has stepped on that cell
      bool marked;     // whether the player thinks this is a mine
};
```

Is used to represent a cell of the mine field.

The number of mines in the mine field should be determined based on the size of the grid provided by the user and the mine density predefined in the code.

To debug your code, the part of the program that runs the game algorithm is commented out. Besides, you can temporarily make all the cells visible so that the display function shows the content of the cells.

Once you make sure that the function is producing a correct mine field, you can make the cells hidden again and uncomment the game algorithm implementation to make sure that the game is running properly.