CS207 Programming Methodology and Abstractions

Fall 2019: 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 14/10/2019 at 2:30 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.

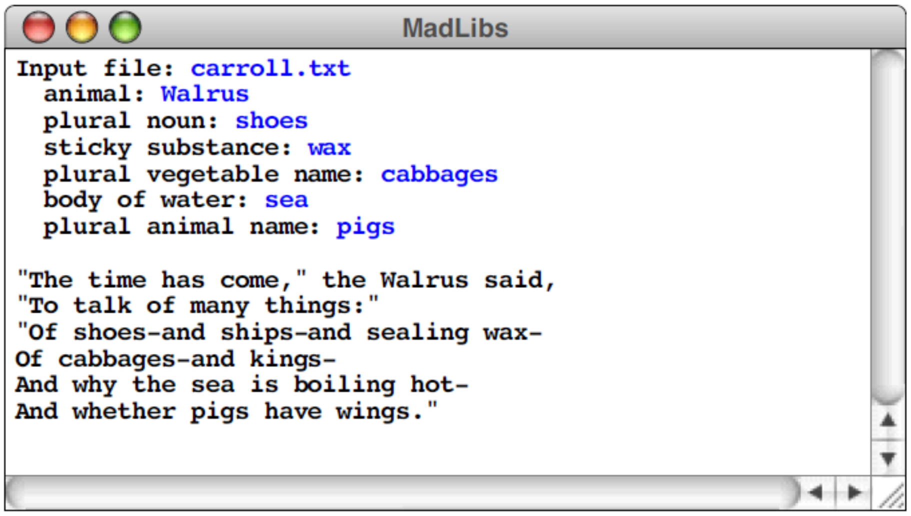
Exercise 1 (50 pts):

In the 1960s, entertainer Steve Allen often played a game called *madlibs* as part of his comedy routine. Allen would ask the audience to supply words that fit specific categories—a verb, an adjective, or a plural noun, for example—and then use these words to fill in blanks in a previously prepared text that he would then read back to the audience. The results were usually nonsense, but often very funny nonetheless.

In this exercise, your task is to write a program that plays *madlibs* with the user. The text for the story comes from a text file that includes occasional placeholders enclosed in angle brackets. For example, suppose the input file carroll.txt contains the following excerpt from Lewis Carroll’s poem “The Walrus and the Carpenter,” with a few key words replaced by placeholders as shown:

```
"The time has come," the <animal> said,
"To talk of many things:"
"Of <plural noun>-and ships-and sealing <sticky substance>-
Of <plural vegetable name>-and kings-
And why the <body of water> is boiling hot-
And whether <plural animal name> have wings.”
```

Your program must prompt the user for the name of a file containing a madlib (sample file caroll.txt attached). Then, read this file and prompt the user for input to fill in the placeholders with new strings. If Carroll himself had used the program, he would likely have obtained the following sample run:



```
Input file: carroll.txt
  animal: Walrus
  plural noun: shoes
  sticky substance: wax
  plural vegetable name: cabbages
  body of water: sea
  plural animal name: pigs

"The time has come," the Walrus said,
"To talk of many things:"
"Of shoes-and ships-and sealing wax-
Of cabbages-and kings-
And why the sea is boiling hot-
And whether pigs have wings."
```

Note that the user must provide all the substitutions before any of the text is displayed. This design makes it impossible to display the output text as you go. The program then must write the output to a file first and then copy the contents of the output file back to the screen.

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 one function which aim is to generate the initial state of the mine field. The objective of the exercise is to come up with the appropriate prototype for the function, implement it and add the corresponding function call in the main function. The role of the function includes:

- Prompting the user and getting the width and height of the mine filed.
- Dynamically allocating a 2D array of cells
- Calculating the number of mines in the mine field
- Randomly placing the mines in the mine field
- Filling the rest of the grid with the information about the number of mines in the 8 neighboring cells.

Note that, besides initializing the main data structure i.e. the mine field, this function should also provide the calling function (main) with the width, height and number of mines. This information is obtained (from user or by calculating it) within the function and is needed for displaying it and running the game algorithm and interface.

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 function, it is recommended that at first you comment out the part of the program that runs the game algorithm. 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 restore the game algorithm implementation to make sure that the game is running properly.