ASSIGNMENT 1

COMP-322B, Winter 2011

Due: Tuesday, February 8, 2011 (23:55)

You **MUST** do this assignment individually. The assignment must be submitted via *myCourses*.

Part 1, Question 1:	20 points
Part 2, Question 1:	30 points
	50 points total

Part 1: Conditionals and Arrays

Question 1: A Game of Tic-Tac-Toe (20 points)

In this question, you will implement a console-based game of tic-tac-toe. *Rules of the game:* Tic-Tac-Toe is played on a 3-by-3 grid with two players. Each player takes a turn to play either an `X` or an `O` on the grid. The first player to get three in a row either diagonally, horizontally or vertically wins the game.

In order to implement this game, we need to consider the following design criteria:

1. How to store the grid
2. How to select a square on the grid
3. How to alternate players' turns and ensure they enter a valid selection
4. How to detect a winner or game over

Hints: You can use a multidimensional array (3-by-3) to store the grid. Each square in the grid may be represented by a number, so that a player can select a square by entering its number. After each turn, your program should check the grid for a winner (if there are three consecutive `X`'s or `O`'s). The game should end when there is a winner or when nine turns have elapsed, whichever occurs first. Loops will come in handy for this question. On the course webpage, you are provided with a file `Tictactoe.cpp` to help you get started.

Sample session:

```
> g++ -Wall Tictactoe.cpp -o tictactoe
> ./tictactoe


1 | 2 | 3
---------
4 | 5 | 6
---------
7 | 8 | 9

Player 1, please enter the square number to place your X: 7
```

```
1 | 2 | 3
---------
4 | 5 | 6
---------
X | 8 | 9

Player 2, please enter the square number to place your O: 5


1 | 2 | 3
---------
4 | O | 6
---------
X | 8 | 9

Player 1, please enter the square number to place your X: 3


1 | 2 | X
---------
4 | O | 6
---------
X | 8 | 9

Player 2, please enter the square number to place your O: 9


1 | 2 | X
---------
4 | O | 6
---------
X | 8 | O

Player 1, please enter the square number to place your X: 1


X | 2 | X
---------
4 | O | 6
---------
X | 8 | O

Player 2, please enter the square number to place your O: 2


X | O | X
---------
4 | O | 6
---------
X | 8 | O

Player 1, please enter the square number to place your X: 4
```

```
X | O | X
---------
X | O | 6
---------
X | 8 | O

Player 1 WINS!
```

Spiritual Growth Question (0 points): Instead of having two human players, suppose that you now want to implement the game so that one person plays against the computer. How would you design an AI which intelligently plays the game? (There are no points awarded for this questions, and it should not be submitted. This is meant to motivate those interested in AI. You need not write an implementation for this question, but think about possible strategies.)

Part 2: Pointers, References and Memory Allocation

Question 1: Music Records (30 points)

In this question, you will be creating a small application which models a music database system. This application will create and maintain an array of `music` records.

Each `music` record will have have two fields: `artist` (of type `string`) and `songTitle` (of type `string`). You will need to use a `struct` to create this record. You will also need to use operations from the C++ `string` library.

This array will be dynamic, in the sense that it can expand as needed. One problem with arrays however is that once the array is created using the `new` operator, its size cannot change. You can use dynamic memory to expand the array as needed. That is, the array will have an initial capacity, but once maximum capacity is reached, you will create a new array of larger size, and copy all entries over. You will handle this case in your `addSong()` function. Also note that if an entry of the array is deleted, you must ensure that all entries remain contiguous. You should consider having a variable which keeps track of how many entries are present in the array.

You will implement the following functions to perform operations on the array:

1. `findSong()`, This function searches the array for the given record. Two music records are considered equivalent if their `artist` and `songTitle` fields are equivalent. If the record is found, the function returns its index, otherwise it returns `-1`.
2. `addSong()`, This function adds a new record to the array. Your array should not contain duplicates. The `addSong()` function should first search the array for an equivalent record using the `findSong()` function. If the record does not already exist, then the new record is added, otherwise, the array must remain unchanged. You must also handle the case where the array has reached full capacity. In this case, you create a new array of size capacity+1, copy existing entries over, and add the new entry.
3. `deleteSong()`, This function searches for the record to be deleted. If it exists, the record is removed, otherwise the array remains unchanged. The array should not have empty spaces in between. All entries should remain contiguous.
4. `printAll()`, This function prints all entries to screen. See sample session below for an example.
5. Your `main()` function will create an array of type `music`, originally of capacity 10. It should display the following menu to the user, and prompt them to enter a choice:

```
-------Menu-------
1. Add a new song
2. Delete a song
3. Display all songs
4. Quit
Enter your choice (1 / 2 / 3 / 4):
```

You may assume the user will enter an integer number and need not handle the case where the user may enter any other type. While the user inputs an invalid number, the menu should keep being displayed. The menu should be re-displayed to the user until they decide to quit the application (by entering choice number 4).

Operations are to be performed on the array according to the user's choice. For choices 1 and 2, you should prompt the user to enter the `artist` and `songTitle`, to create the record to feed to the add and delete functions.

You need not create a class for this problem, as this is a non-OOP problem. The file `MusicRecords.cpp` is provided on the webpage as a starting point.

Sample Session:

```
> g++ -Wall MusicRecords.cpp -o music
> ./music

-------Menu-------
1. Add a new song
2. Delete a song
3. Display all songs
4. Quit
Enter your choice: (1 / 2 / 3 / 4): 1

Enter artist:
Dream Theater
Enter song title:
The Glass Prison
Song successfully added!

-------Menu-------
1. Add a new song
2. Delete a song
3. Display all songs
4. Quit

Enter your choice: (1 / 2 / 3 / 4): 1
Enter artist:
Scorpions
Enter song title:
Wind of Change
Song successfully added!

-------Menu-------
1. Add a new song
2. Delete a song
3. Display all songs
4. Quit
Enter your choice: (1 / 2 / 3 / 4): 2
Enter artist:
Scorpions
Enter song title:
Wind of Change
Song Deleted!

-------Menu-------
1. Add a new song
2. Delete a song
3. Display all songs
4. Quit
Enter your choice: (1 / 2 / 3 / 4): 3

Record 1
Artist: Dream Theater
Song Title: The Glass Prison

-------Menu-------
1. Add a new song
2. Delete a song
3. Display all songs
4. Quit
Enter your choice: (1 / 2 / 3 / 4): 1

Enter artist:
Dream Theater
Enter song title:
The Glass Prison
Song already exists!

-------Menu-------
1. Add a new song
2. Delete a song
3. Display all songs
4. Quit
Enter your choice: (1 / 2 / 3 / 4): 4
Goodbye
```

What To Submit

```
Tictactoe.cpp
MusicRecords.cpp
```