Cairo University
Faculty of Computers and Information
Computer Science Department

Programming-1 CS112
2017/2018 First Term

Assignment 2

Prof. Abeer ElKorany
Dr. Amin Allam

1 [Task 1] Students database

⇒ Create a student structure as follows:

```
1  const int MAX_NAME=14; // Assume a name contains at most 14 characters
2  enum Gender{MALE,FEMALE};
3  struct Date{int year; int month; int day;};
4
5  struct Student
6  {
7     int    ID;
8     char   first_name[MAX_NAME+1];
9     char   last_name[MAX_NAME+1];
10    Date   birth_date;
11    Gender gender;
12    double GPA;
13 };
```

⇒ Define a function `InputStudent()` which takes a student data from the user and saves it into a `Student` object.
⇒ Define a function `OutputStudent()` which prints a student data in one line.

```
1  void InputStudent(Student* s);   // s is a pointer to one object
2  void OutputStudent(Student* s);
```

Each student data should be output in *one line* in the following order:
`ID first_name last_name birth_date gender GPA`
The following is an example of student data input and output:

```
20090111 Aly Ahmed 26/5/1992 Male 3.4
```

⇒ Define a function `InputAllStudents()` which inputs `n` students from the user and saves them in a dynamic array which was previously allocated in `main()`.
⇒ Define a function `OutputAllStudents()` which prints all student data, each student in one line.

```
1  void InputAllStudent(Student* s, int n);    // s is a pointer to dynamic
2  void OutputAllStudent(Student* s, int n);   //  array of n objects
```

⇒ Define a function `SortStudents()` that sorts `n` students. The function should take a function pointer `LessThan()` as a parameter. The function `LessThan()` takes 2 student pointers and returns `true` if the first student should come before the second student after sorting.

```
1 void SortStudents(Student* s, int n,
2                   bool (*LessThan)(Student*,Student*));
```

⇒ Define a function `BirthLess()` that takes 2 student pointers and returns `true` if the first student is born before the second student. Use it as a parameter to `SortStudent()` in order to sort them by increasing birth date.
⇒ Define a function `GPAGreater()` that takes 2 student pointers and returns `true` if the GPA of the first student is greater than the GPA of the second student. Use it as a parameter to `SortStudent()` in order to sort them by decreasing GPA.

```
1 bool BirthLess(Student* a, Student* b);
2 bool GPAGreater(Student* a, Student* b);
```

⇒ Define a function `SearchStudentID()` that takes `n` students and a student ID as parameters and returns a pointer to a student having this ID or `0` if not found.
⇒ Define a function `SearchStudentFirstName()` that takes `n` students and a C-string as parameters and returns a pointer to any student having this first name or `0` if not found.

```
1 Student* SearchStudentID(Student* s, int n, int ID);
2 Student* SearchStudentFirstName(Student* s, int n, char* name);
```

⇒ The program should start by taking the number of students `n` from the user. The program should create a dynamic array of `n` students.
⇒ Then, the program should take the full data of each of the `n` students from the user.
⇒ Then, the program should output a list of choices for the user as follows:

```
1) Output all students data.
2) Sort students by increasing birth date.
3) Sort students by decreasing GPA.
4) Search students by ID.
5) Search students by first name.
6) Exit the program.
```

⇒ If the user inputs `1`, the program should output the full data of the `n` students, each student in one line, then reprints the list of choices.
⇒ If the user inputs `2`, the program should sort the `n` students by increasing birth date then outputs all sorted students, then reprints the list of choices.
⇒ If the user inputs `3`, the program should sort the `n` students by decreasing GPA then outputs all sorted students, then reprints the list of choices.
⇒ If the user inputs `4`, the program should take integer from the user, searches for a student having this ID, prints the student full data if it exists, then reprints the list of choices.
⇒ If the user inputs `5`, the program should take a C-string from the user, searches for a student having this first name, prints the student full data if it exists, then reprints the list of choices.
⇒ If the user inputs `6`, the program should release the dynamic array and exit.

2 [Task 2] Connect four game

⇒ Write a two player connect 4 game. The board consists of 6 rows and 7 columns. Each player chooses a column from 0 to 6 to insert his tile. The player who arranges 4 tiles in a row/col/diagonal wins. Refer to https://en.wikipedia.org/wiki/Connect_Four for more details about the game.

⇒ The board data should be a 2D array of characters, which assigns to each board cell a space character `' '` if it is empty, an `'X'` if it contains a tile from first player, or an `'O'` if it contains a tile from second player.

```
1 char board[6][7];    // each cell is either 'X', 'O', or ' '
```

⇒ Write a function `IsWinning()` that takes the board as a parameter and returns `'X'` if the first player wins, `'O'` if the second player wins, or `' '` if no one wins.

```
1 char IsWinning(char board[6][7]);
```

⇒ Write a function `Insert()` that takes the board, the player tile (`'X'` or `'O'`), and the column where the player chooses to insert his tile (from `0` to `6`) as parameters. The function should update the tile by modifying the appropriate empty cell to contain this tile and returning `true`, or returning `false` if the column does not contain empty cells.

```
1 bool Insert(char board[6][7], char player, int column);
```

⇒ Write a function `Print()` that prints the current full board on screen.

```
1 void Print(char board[6][7]);
```

⇒ Write a `main()` function that takes the column number (from `0` to `6`) from first player, inserts `'X'` in the appropriate place, then prints the full board. After that, it takes the column number from second player and inserts `'O'` in the appropriate place, then prints the full board. If a player selects a full column, the program requests from the player to enter the column number again until he enters a valid column. If a player wins, the program prints the winner and exits.

3 Delivery Notes

- This is a group assignment of 2 members (at most) and the members should be from the same lab (i.e. G1&G2 can work together).
- Both students should work and fully understand everything in the code.
- Due date is on Saturday 9/12/2017 until 11:59 pm (i.e., until midnight).
- Assignment will be discussed in labs starting from 10/12/2017.
- No late submission is allowed.
- Submission will be through the Acadox: `http://www.acadox.com/class/48405`
- No submission through e-mails.
- For each task you will develop a .cpp file that should include a block comment containing students IDs and names. These files should be named task1.cpp and task2.cpp. Then put these 2 files in a folder named:
  `Prog1_SecondAssignment_FirstStudentID&SecondStudentID`
  and compress them to a .zip file with the same folder name. The compressed file would be the file to be delivered.
- Failing to abide by the naming conventions of the file, would result in a ZERO for both team members.
- For students who do their assignments on shared machines (e.g., on the FCI labs machines or the students residence machines), please make sure to permanently delete your files from the shared machines after you finish your work, so that no other student would take your files and submit it under their names.
- In case of Cheating if the assignment is from (N) marks then the one who have committed cheating will get (-N).
- You have to write clean code and follow a good coding style