The British University In Egypt الجامعة البريطانية في مصر Informatics and Computer Science	**19CSCI01P** **Assignment instead of two lab tests** **2019 / 2020**	
Module Title: **Introduction to Programming and Problem Solving**		
Module Leader: **Dr. Mostafa Salama**		Semester **Two**
Assessment Weight **50% of the total module mark**	Due Date **May**	

The total mark of the assignment is 60 Marks and it carries out 50% of the total mark of the course. ***All submissions will be passed by an accurate plagiarism checker.***

Project Objective

The main objective of this assignment is to test your understanding of the main concepts of programming.

Individual Work

Each student should submit the assignment including his name, ID, and group.

Timeline

The response of the assignment should be submitted within 2 weeks of publishing the assignment.

Submission Contents

The project text code with the proper comments, name the c++ file as "*PrepAssignmentCode.cpp*" and the file of data *"PrepAssignment.txt"*. Use the same variable names as shown in the code specifications, that has following format ***students***.

Feedback

The marks will be published on the E-Learning including the feedback on your deliverables.

1. ***Program description***:

   Write a C++ program that contains an array of 20 ***students****. Each* ***student*** *has a set of information {****name****(string),* ***id****(integer),* ***gender****(boolean),* ***major****(pointer to a string),* ***grades****(array of 5 doubles, each represents a grade out of 100 of single module), and* ***year****(enum* ***yearofStudy****:* ***prep****,* ***year1****,* ***year2****,* ***year3****)}.*

   - *The first time this program runs, the content of* ***students****' array is received from the user, then save them in a file of name "****PrepAssignment.txt****". One student is saved at a time to file.*
   - *Otherwise, the program loads the data in the* ***students****' array from the file "****PrepAssignment.txt****".*
   - *Then the program shows the following list of options to the user.*
     1) *Delete a student,*
     2) *Edit the information of a student,*
     3) *Add a new student if the list is not full,*
     4) *Print the number of students in a specific major,*
     5) *Print the number of students in a specific year,*
     6) *Exit*
   - *After performing the selected option, the program reprints the list again for further selection. If the user selected the "Exit" option, the program re-saves to the "PrepAssignment.txt" file.*

2. ***Code specification:***
   - ***Class student****:* **[10 Marks]**
     1) *Define a class "****student****" of member variables "****name****", "****id****", "****gender****", "****major****", "****grades****", and "****year****".*
     2) *All the member variables of class "****student****" are private.* **[1 marks]**
     3) *The class contains a function that returns the average grade of the student, named as "****getAverageGrade()****".* **[10 Marks]**
   - ***Main function****:*
     4) *Initialized 4 strings,* ***major1****="cs",* ***major2****="is",* ***major3****="it", and* ***major4****="se".*
     5) *Define an array of students "****students****"*
     6) *The* **student.major** *member variable is a pointer that points to any of the 4 strings (****major1****,****2****,****3****,****4****). The value of the* ***student.year*** *member variable is dependent enumeration* ***yearofStudy****.*
     7) *Check if the file "PrepAssignment.txt" contains data or not.*
     8) *If the file "PrepAssignment.txt" is empty, the main function gets the students information from the user and save it in the* ***students****' array. Then save this information in the PrepAssignment.txt.*

9) *If the file "PrepAssignment.txt" contains data, then the main function loads the data from the file and save it in the* ***students****' array.*

10) *Then the main function prints the above list of options to the user and ask him to select one of these options.*

11) *If the user selects option number 1, the program asks the user to enter the* ***id*** *of the student. Then the main function calls a function* ***deleteStudent****(****students****,* ***id****).*

12) *If the user selects option number 2, the program asks the user to enter the id of the student. Then the main function calls a function* ***editStudent****(****students****,* ***id****).*

13) *If the user selects option number 3, the program checks if the array is not full. Then the main function calls a function* ***addStudent****(****students****). If the file is full, the program apologizes to the user from not being able to process this task.*

14) *If the user selects option number 4, the program asks the user to enter which major he is requiring. Then program calls function* **countStudents**(*). Then returned value from this function is printed.*

15) *If the user selects option number 5, the program asks the user to enter which year he is requiring. Then program calls function* **countStudents**(*). Then returned value from this function is printed.*

16) *If the user selects any option other-than "****Exit****", the program does the required option then asks the user to enter a new option. The program continues asking the user to enter a new option until the user selects the "****Exit****" option.*

17) *If the user selects the "****Exit****" option, the program will save the array elements in the file.*

deleteStudent() function **[10 Marks]**

18) *This function contains two parameters, the first parameter is the students array, and the second parameter is the id of the student that is required to be deleted by the user.*

19) *The function searches for the student's id in the "****students****" array. If the id exists, the program deletes the student of this id.*

20) *Then the function replaces the empty cell of the deleted student by the next student in the array, like shifting the next student to left. The program continues shifting until all next students are moved by one cell to the left.*

For example, consider the following list

st. 1	**st. 2**	**st. 3**	**st. 4**	**st. 5**	**st. 6**	**st. 7**	**st. 8**	**st. 9**

Delete student "st.4"
The list becomes as following:

st. 1	**st. 2**	**st. 3**	**st. 5**	**st. 6**	**st. 7**	**st. 8**	**st. 9**	

editStudent() function **[10 Marks]**

21) *This function contains two parameters, the first parameter is the students array, and the second parameter is the id of the student that is required to be edited by the user.*
22) *The function searches for the student's id in the "**students**" array.*
23) *If the id exists, the function displays the information of this student, then allows the re-enter name and to re-enter the year, and to re-enter the gender, and to re-enter the major. Then the function asks if need to exit this option to main list or edit another student.*
24) *If the id does not exists, the function prints "missing id" then asks the user to re-enter a new id to check or to exit this option.*
25) *The function continues in a loop until the user types an existing id.*

addStudent(students, position) function **[10 Marks]**

26) *This function contains the students array as a parameter.*
27) *The function gets the information of the **student** from the user.*
28) *if the **position** parameter is -1, then the function adds this student to the end of the array,*
29) *if the **position** parameter is any number less than 20, then the function shifts all the array elements starting from the index to the right. Then the function adds this student to the array cell of index = **position**.*

St.1	**St.2**	**St.3**	**St.4**	**St.5**	**St.6**	**St.7**	**..**	**..**	**..**

position = 4

St.1	**St.2**	**St.3**		**St.4**	**St.5**	**St.6**	**St.7**	**..**	**..**

Insert the new student (St.N) in position 4

St.1	**St.2**	**St.3**	***St.N***	**St.4**	**St.5**	**St.6**	**St.7**	**..**	**..**

countStudents(students, majorP, yearP) function **[10 Marks]**

30) *This function checks the **majorP** parameter, if it is pointing to a correct string and not **NULL**. Then it counts the number of the students in the **students** array whose **major** is the same as the parameter **majorP**.*
31) *If the **majorP** parameter is **NULL**, this function checks the **yearP** parameter. If its value within the range of the enumeration (**yearofStudy**) , then it counts the number of the students in the **students** array whose **year** is the same as the parameter **yearP**.*
32) *The function returns an integer that reflects the required number of students.*

Good Luck