COSC1337 - Programming Fundamentals II **Spring 24**
Programming Assignment 3 **Due Date: May 5th @ 11:59 pm**

This assignment covers the following topics:

- Classes and Objects
- Aggregation
- 2d array of objects

Assignment Description:

A summer STEM camp hosts a number of students majoring in COSC, MATH, and BIOL. On the first day of the camp, the students are asked to gather in an amphitheater with the seating arrangement of 10 rows of 12 seats. They will be asked to sign in with their information and take a seat. We have the sign-in sheet with the information of ALL attendees/students in an input file called *studentData.txt*. The information in this file leads us to designing the class Student whose header file with ALL inline functions can be downloaded from Canvas.

The requirements of this assignment is broken in the following tasks each with detailed outlines of their requirements. Complete them in the order they appear below:

Task I. Give the code to simulate the amphitheater's seating arrangement. Each seat will have a `Student` object whose information must be populated via the file. The following gives a sample of this input file.

```
B91 BIOL 3.00
M28 MATH 3.90
B10 BIOL 3.00
. . .
```

Note that we do not know how many students' information is in the file or how many students we have per major. But we DO know that the maximum capacity of the camp facilities is 120 students, which is exactly the number of seats in the amphitheater. But the actual number of students in the camp is unknown. We know that the students per major have the maximum of one-third of the total, meaning we could have the maximum of 40 students per major. But the actual population of students per major is also unknown.
When you read information for each student and instantiate and populate an object for them, randomly "seat" them in your "amphitheater" data structure which is a 2d array of type `Student` called `seatingChart`. This random population should be done by randomly choosing a number from the range ***`0-(ROWS*COLS-1)`*** inclusive. The mapping of this single number into a (row, column) coordinates MUST be computed. We have done such computation at the beginning of the semester, Remember? You need to make sure that you are not choosing a single slot multiple times causing objects to be overwritten.

When done processing the file, display the seating map of the amphitheater by displaying the students' ID's for the occupied seats and "- -" for unoccupied ones via call to a function with the following prototype:

```
void displaySeating(Student [][COLS], int);
```

```
======================== Task I ========================
The original Random Seating Map:

B16  --   --   --   --   C17  C57  C37  B34  --   B94  --
B85  --   B45  B49  B88  M32  --   M58  --   B27  C54  C30
C63  M62  --   --   --   B98  M93  --   M82  --   M25  B40
M43  M81  C61  M15  C71  --   B51  C43  M94  B48  --   B37
M95  --   C32  B91  C94  --   --   M91  B14  --   --   --
M35  C27  M50  B59  B21  C11  C49  --   --   B77  C10  M70
M55  B65  M31  C28  --   M49  --   --   M68  --   C12  M97
M57  --   B71  C89  M21  --   C24  C23  --   --   M71  M90
--   M69  C69  --   C46  C88  C96  --   --   M28  B76  C16
--   B86  M30  M98  M72  --   B19  --   M11  C55  --   B10
```

Task II. The camp coordinators would like to seat students of the same major in three separate sections of the seating area. In a 10 by 12 seating chart, the first subsection of 4 by 10 will be dedicated to students in COSC, the adjacent subsection of 4 by 10 for the MATH majors and the remaining section will seat the students from BIOL. The simplest way to go about this task is to go through the original seating chart and populate three vectors corresponding to students of three majors. We would like to have these major-based vectors to be members of an array of 3 vectors of `Student` objects. Declare this array of size 3 and call it `majorVectorList`. Next step is to iterate over the original seating array and add each `Student` object to their respective vector stored in one of the three slots in `majorVectorList`. Notice that we could use an `enum` type for this part, but for the sake of brevity of the assignment I did not make it a requirement. You can use hardcoding to store COSC majors in index 0 of this array of vectors, MATH majors will be in index 1, and index 2 will have a vector of BIOL majors all with the help of the following functions that you MUST use:

```
int stringToIndex( string);
string indexToString ( int );
```

Now that we have three vectors of students in each major, you MUST use a function with the following prototype to map them in different sections of the seating chart:

```
void groupMajors(Student [][COLS], int, vector <Student>, int, int );
```

This function receives the seating array, its number of rows, a vector of `Student` objects in a given major, the range of columns for the subsection they will be seated based on this new re-arrangement. You must call this function in a 3-iteration loop and pass each vector in the `majorVectorList` to it, but then you MUST pass to them the beginning and the end column for each subsection. Think about this: the leftmost subsection will have the range of columns in 0-3 inclusive, next will be 4-7 inclusive and the last will be 8-11. Think of the pattern and code accordingly.

Also note that each function call will store ALL the students of that major in their respective subsection of the seating array. When done with populating each section with their respective vector elements, you need to "clear" the contents of the remaining slots in the sections. This is so that you won't have the `Student` objects who are now supposed to be reseated to appear in their previous seats. Declare a temporary `Student` object as follows and copy that to the remaining spots in the corresponding subsections across the seating array for each major.

```
Student clearObject("--", "", 0.0);
```

To show your work for this task, display the content of the seating array via a call to the `displaySeating` function. Use the following sample run of this segment as a reference:

```
========================= Task II =========================
The Reconfigured Seating Map:

C17  C57  C37  C54  M32  M58  M62  M93  B16  B34  B94  B85
C30  C63  C61  C71  M82  M25  M43  M81  B45  B49  B88  B27
C43  C32  C94  C27  M15  M94  M95  M91  B98  B40  B51  B48
C11  C49  C10  C28  M35  M50  M70  M55  B37  B91  B14  B59
C12  C89  C24  C23  M31  M49  M68  M97  B21  B77  B65  B71
C69  C46  C88  C96  M57  M21  M71  M90  B76  B86  B19  B10
C16  C55  --   --   M69  M28  M30  M98  --   --   --   --
--   --   --   --   M72  M11  --   --   --   --   --   --
--   --   --   --   --   --   --   --   --   --   --   --
--   --   --   --   --   --   --   --   --   --   --   --
```

Task III. The re-arrangement/re-seating of the students in three subsections will help the coordinators to address different majors in person, but they will create challenges for us code-wise. In order to help ourselves in navigating this new seating "landscape" in code, we declare three vectors of pointers to `Student`. You did not think that I let you end the semester with no pointers making an appearance in the last assignment, did you?!

This process is much like populating the `majorVectorList` array, but we are populating what you should call the `sPtrVectorList` array of size 3.

Iterate over the seating array, and store the addresses of students in different majors in the proper vectors in the `sPtrVectorList` array. Make sure to only store the addresses of the occupied slots in the seating array.

So as of now in our program we have the following:

- A seating 2d array with students sitting in three sections: COSC, MATH, and BIOL, from left to right, with the unoccupied slots storing `clearObject`.
- The `majorVectorList` array that contains 3 vectors of `Student` objects grouped in each index, 0 corresponding to COSC, 1 corresponding to MATH, and 2 corresponding to BIOL majors.
- The `sPtrVectorList` array that contains three vectors of pointers to `Student` objects with the exact storage pattern of the `majorVectorList` array.

Nothing to show here! This prepares our program to tackle the next task.

Task IV. You find the *Team.h* file in Canvas, where you need to complete all the inline functions. The description of these functions are given as comments throughout the file. This task prompts the user who is assumed to be a faculty member for their full name, the name of the project and the number of potential members/students of this project. Study the following description of this task to instantiate an object of type `Team` and use its functions and attributes in completing the requirements of this task:

In order to pick students for an advanced science project, students need to take a qualifying exam. The students with the top highest scores in this exam will be chosen for this project. There is no limit on how many of which major will be chosen, the sole criteria is the test score. After students are chosen for this project, they will be asked to leave the amphitheater since they will not be considered for any other camp projects due to the demanding schedule of this particular science project. We would like to simulate this process via randomly selecting students. The random process MUST be based on a 2-step selection involving ONLY The `sPtrVectorList` array. First, pick a random number in 0-2 inclusive, this will choose a major. Depending on this random number, the next random number will be based on the size of the vector in that index. We use these two random numbers to reach to the pointer in that slot and copy the `Student` object in the dynamic array that is an attribute of the `Team` class. You must remove the chosen students from the seating chart as they will be asked to leave the amphitheater. To be able to visually check the accuracy of your work for this task we do the removal process through overwriting the `Student` objects who are selected for the Science Project with the following object:

```
Student spStudent("SP", "", 0.0);
```

Now that you know about the process of student selection. We need to properly instantiate a Team object and use its functions to simulate this process and add qualifying students in this Team objects' `members` attributes. After prompting the user for their full name, the name of their project and the total number of team members, you are ready to instantiate a `Team` object called `sciTeam`. As you find in the `Team`'s header file , the `Team`'s constructor takes care of initialization of objects including dynamically allocating its `Student` array attribute based on the size specified by the user. Follow the strategy outlined above in randomly choosing students and use the proper member functions for `Team` objects, to simulate the process for this task.

Call the `displayMemberIDs` function for `sciTeam` to display the members of this team. Also call the `displaySeating` as a proof for your work required for this task.

The following is a sample run of this task. Keep in mind that this is a random process and your result may be different:

```
Registering your project . . .

Please enter the faculty's full name:    Emma Nixon
Greetings Dr. Emma Nixon
Please enter the name/ID of your project:      Science Project
How many students are you considering for the Science Project?  20
Instantiating your Team ...


======================== Task IV =========================
C17  C57  SP   C54  SP   M58  M62  M93  B16  B34  B94  B85
C30  SP   SP   C71  M82  M25  SP   M81  SP   SP   B88  SP
C43  C32  C94  C27  SP   M94  M95  M91  SP   B40  B51  B48
SP   C49  C10  C28  SP   M50  M70  M55  B37  B91  B14  B59
C12  C89  SP   C23  M31  SP   M68  M97  B21  B77  B65  B71
SP   C46  C88  C96  M57  SP   M71  M90  B76  SP   SP   B10
SP   SP   --   --   M69  M28  M30  M98  --   --   --   --
--   --   --   --   M72  M11  --   --   --   --   --   --
--   --   --   --   --   --   --   --   --   --   --   --
--   --   --   --   --   --   --   --   --   --   --   --


Members of the team Science Project:
C61  B98  M49  B19  C55  M21  M35  B27  C69  C63  C37  M15  B45  M32  B86  C16  B49  M43  C11  C24
```

Task V. Give the code to report the number of students chosen for the Science Project, per major. You should implement this task as a tally problem. Use the conversion functions: `stringToIndex` and `indexToString` you defined earlier to help with this task. Note that this segment of code needs access to the members of the `Team` object. You have a method that gives you a safe access to this data structure that you can use here.

```
====================== Task V ========================
The major tally of the team members:
COSC: 8
MATH: 6
BIOL: 6
```

Task VI. Another faculty would like to work with the Science Project team in a different project that studies the effectiveness of camp activities in academic and social development of the students. In order to give this faculty access to this team we would like to make a clone of the `sciTeam` via call to the copy constructor that you have defined for the `Team` class. Call this new `Team` object `socialSciTeam`. Prompt the user for the name of the faculty as well as the name of the new project, so that you could use the set methods of `Team` object to update these two attributes of this object. Call the `displayTeamInfo` method to display the full information of this new `Team` object.

```
Please enter your full name:       Paul Adams
Please enter the name of your project:   Social Science Project

====================== Task VI ========================
The information of the new team:
Team:              Social Science Project
Team Advisor:      Paul Adams
The Number of Members: 20

Members of the team Social Science Project:
C61  B98  M49  B19  C55  M21  M35  B27  C69  C63  C37  M15  B45  M32  B86  C16  B49  M43  C11  C24
```

Task VII. A student on `sciTeam` can no longer be a member of this team due to scheduling conflict, however they do remain in `socialSciTeam`. In order to remove this student from the `sciTeam` object, we first should prompt the user for their ID. So, display only the IDs of the team members of `sciTeam`; prompt the user to input their ID from the list. The removal process of a `Student` object will mean removing it from the member list of the Team object AND returning it back to their original spot in the seating array. We MUST take care of their return to the seating chart before permanently removing them from the `Team` object. In order to test ALL the information and data structures that we already have in our program, this is how we go about finding this student, returning them to the seating chart and removing them from the Team object:

Pass the ID to the proper member function of the `Team` class that finds `Student` objects based on their ID. Use the object that it returns to get to their major. Next, use the major of the student to hone in the index in the `majorVectorList` array, and subsequently use their ID to look up their object stored in the proper vector in the `majorVectorList` array. When the target object is found, use the exact index in which you found the `Student` object, this time in the `sPtrVectorList` array, to get to the pointer that used to point to the slot this object was previously "seated" in the seating chart. Remember the `majorVectorList` array and the `sPtrVectorList` array are in logical parallel index-by-index. In the same indexes in both of these array, one stores the actual object and the other one has the address of the location in which the object is seated or used to be seated in the seating chart. Use this pointer to its previous seating slot in the seating chart to RE-ASSIGN them in the seating chart. Display the seating chart to verify the accuracy of your work for this task.

Now it is safe to call the proper remove function to remove the student form the list of members by passing its ID to it. Use the `Team` object to display the members to verify the successful removal of the student as well as a report of the updated size of the team.

```
======================= Task VII ========================
We have received your request for removing a student from your project team ...

Please choose the ID of the student to be removed from the following list:
Members of the team Science Project:
C61  B98  M49  B19  C55  M21  M35  B27  C69  C63  C37  M15  B45  M32  B86  C16  B49  M43  C11  C24


Enter the ID:    C55

C55 is returned to their seat:

C17  C57  SP   C54  SP   M58  M62  M93  B16  B34  B94  B85
C30  SP   SP   C71  M82  M25  SP   M81  SP   SP   B88  SP
C43  C32  C94  C27  SP   M94  M95  M91  SP   B40  B51  B48
SP   C49  C10  C28  SP   M50  M70  M55  B37  B91  B14  B59
C12  C89  SP   C23  M31  SP   M68  M97  B21  B77  B65  B71
SP   C46  C88  C96  M57  SP   M71  M90  B76  SP   SP   B10
SP   C55  --   --   M69  M28  M30  M98  --   --   --   --
--   --   --   --   M72  M11  --   --   --   --   --   --
--   --   --   --   --   --   --   --   --   --   --   --
--   --   --   --   --   --   --   --   --   --   --   --


The updated information of the team:
Team:               Science Project
Team Advisor:       Emma Nixon
The Number of Members: 19

Members of the team Science Project:
C61  B98  M49  B19  M21  M35  B27  C69  C63  C37  M15  B45  M32  B86  C16  B49  M43  C11  C24
```

Task VIII. Give the segment of code to display the count of `Team` objects.

```
======================= Task VIII ========================
The number of Team objects: 2
```

Task IX. ***Bonus Task*** - We would like to find the gpa tally of the members of the Social Science Project. This means to have information about how many of what GPAs are found among the members of this group of students. You are free to come up with any plan you can make this work. One challenge is that we are dealing with floating point values to be tallied. Come up with a plan to make this work and display the result on the screen to show your work for this task.

```
======================= Task IX ========================
The list of GPAs for members of the Social Science Project:
3.30 2.80 2.90 2.90 3.40 2.80 3.50 3.70 3.60 2.90 3.70 3.40 3.50 3.50 3.30 3.90 3.60 3.30 3.00
2.80
```

```
The GPA tally report:

2.80       3
2.90       3
3.00       1
3.30       3
3.40       2
3.50       3
3.60       2
3.70       2
3.90       1
```

Happy Programming!

- *Make sure you submit a program that compiles. An incomplete program that compiles could possibly earn partial points, a non-compiling program will be considered void.*
- *Submit your .cpp file on Canvas by the due date. Submitting wrong files will result in grade of zero.*
- *This is an individual work, a partially or fully identical submissions will earn a grade of zero for all involved.*