## CS 1430 - Program 5 Due Date: Tuesday Dec. 11, 2018 10:00 p.m. Grace Date: Friday Dec. 14, 2018 10:00 p.m. 30 Points

## Objects/Classes: Include enumerations and arrays

## Program Description

You will work in a team of two to write a program to maintain the student records for two teachers. Your program will read and process commands from the standard input until end-of-file. Commands consist of teacher ID, followed by an action command and possibly followed by parameters, with one or more spaces between each item. The commands you are to implement are of the form:

teacher\_id A student\_name strength\_level

teacher\_id R student\_name

teacher\_id P

teacher\_id UQ student\_name student\_quiz\_grade

teacher\_id UT student\_name student\_test\_grade

teacher\_id UH student\_name student\_homework\_grade

teacher\_id UF student\_name student\_finalexam\_grade

1. A – Add a student to the specified teacher’s roster. First check if the student already exists and if it does, print a message. Otherwise, next check if the teacher’s roster is "full" (max. 5 students per teacher) and if it is, print the message: “Can’t handle any more students”. Otherwise, add the new student (name and strength level) with a grades all set to zero. Strength level are characters as follows: M- REMEDIAL, V- ADVANCE, G-REGULAR.
2. R – Remove a specified student from the specified teacher’s roster. First check if the student exists and if it doesn't, print the message: “Can’t remove student student\_name – Student doesn’t exist”. Otherwise, remove the student and print the message: “Record removed for student\_name”.
3. P – Print all students’ information in the specified teacher’s roster. Print out the teacher’s ID, student’s name, strength level, quiz grade, test grade, homework grade, final exam and calculated weighted current grade.
4. UQ – Update specified student’s quiz grade in the specified teacher’s roster.
5. UT – Update specified student’s test grade in the specified teacher’s roster.
6. UH – Update specified student’s homework grade in the specified teacher’s roster.
7. UF – Update specified student’s final exam grade in the specified teacher’s roster.

You don't need to check for bad commands. Assume all commands are correct. All commands will be capital letters. Assume all the parameters are correct. Assume grades to be updated are between 0 and 100.

At the end of the program display the message: “Normal Termination of Teacher Roster Program.”.

See the sample output for the exact wording and spacing for the output.

**Requirements:**

You must implement a **Teacher** class that encapsulates the class roster for one teacher. You are required to have the following **private** data members, no more, no less. The first seven are parallel arrays to hold the teacher’s students’ data. You can't have any method that returns the **raw** private data, either in whole or part, or any method that violates the spirit of data encapsulation and data hiding.

string students[MAX\_STUDENTS];

StrengthLevel level[MAX\_STUDENTS];

int quiz[MAX\_STUDENTS];

int test[MAX\_STUDENTS];

int homework[MAX\_STUDENTS];

int finalExam[MAX\_STUDENTS];

float weightedScore[MAX\_STUDENTS];

int numStudents;

You will have a default constructor that initializes numStudents to 0. You must have and properly use the following **private** methods. **You will lose -4** for each if you don't.

int Find(string student);

void UpdateStudentInfo(int index, char score\_type, int score);

The Find method is similar to the other Find methods we have done: returns the index of where it finds the student in the students array, -1 otherwise. The UpdateStudentInfo method uses score to update the specified quiz, test, homework or final exam (distinguished by score\_type) for the student at the specified index. It must use a switch statement with a default: **-3 if it doesn't**. Neither method is allowed to read or write anything.

You must also have several good public methods to perform the operations on a Teacher object. You will probably have methods such as AddStudent, RemoveStudent, etc. No method of Teacher can take the ID as a parameter or use an ID in any way. Some points for a bad or missing Teacher class:

* -25 if you don't have a Teacher class
* -20 if you don't have the private data members exactly as specified
* -20 if there are any public data members
* -15 if you add any other private data members
* -15 if any method returns the **raw** data, either in whole or part
* -12 if any method takes an ID as a parameter or reads the ID or uses ID in any way

Besides the Teacher class requirements, you must adhere to the following additional requirements:

1. Have constants for following:

* const int MAX\_STUDENTS = 5;
* 15 for the student name and 5 for teacher’s ID field width when printing out the student info for the P commands
* 10 for student strength level and all grades field width when printing out the student info for the P commands
* Weights for each strength level and grades as follows:
  + REMEDIAL:

const float TEST\_WEIGHT\_REM = 0.20f;

const float FINAL\_WEIGHT\_REM = 0.30f;

const float QUIZ\_WEIGHT\_REM = 0.20f;

const float HOMEWORK\_WEIGHT\_REM = 0.30f;

* + REGULAR:

const float TEST\_WEIGHT\_REG = 0.25f;

const float FINAL\_WEIGHT\_REG = 0.35f;

const float QUIZ\_WEIGHT\_REG = 0.15f;

const float HOMEWORK\_WEIGHT\_REG = 0.25f;

* + ADVANCE:

const float TEST\_WEIGHT\_ADV = 0.30f;

const float FINAL\_WEIGHT\_ADV = 0.40f;

const float QUIZ\_WEIGHT\_ADV = 0.10f;

const float HOMEWORK\_WEIGHT\_ADV = 0.20f;

* Teacher ID constants, such as:
  + const string ID\_TEACHER1 = “T1”;
  + const string ID\_TEACHER2 = “T2”;

You do **not** need to make constants for the action commands.

1. You must have and use the following enum in all appropriate places (up to **-6 if you don't**):

enum StrenghLevel { REMEDIAL, REGULAR, ADVANCE };

1. Declare 2 Teacher objects in main such as: teacher1, teacher2;
2. You must show the weighted grade with two decimal places. Put this near the start of main:

cout << fixed << showpoint << setprecision(2) << endl;

1. So that everyone's output matches the Grader output:
   1. When Adding a new Student, add it to the end of the arrays
   2. When Removing a Student, maintain the same order. That means "shoving up" the remaining accounts.
2. Main can be the only place that reads and uses a Teacher ID. **-12 if you violate this**.
3. You must have and use the function (it's not a method) with the following prototype:

void ProcessCommandForTeacher(Teacher & teacher);

This will be called from main with the appropriate Teacher object after main has read and "processed" the ID. It will read and generically process the specified command for a Teacher. It will call the appropriate Teacher method based on the command.

**It must use a switch statement to process the command after reading it.**

The switch will look similar to (assuming command is a char variable into which you read):

switch (command)

{

case 'A':

teacher.AddStudent();

break;

. . . .

Note that the above assumes that the AddStudent method reads the input data. That is **not** a requirement. This function could read the input data and call an AddStudent method that has parameters.

* -6 if you don’t have this function with the exact same header/prototype
* -4 if it doesn't use a switch statement

1. The method that processes the weighted grade calculation must use a switch statement for the strength levels. **-3 if it doesn't**.

1. Recall you must adhere to the group requirements file comment block: **-3 if you don't.**
2. As always, follow the programming ground rules. This includes good modularity – see programming ground rules under “Modularity.” You must have a good decomposition. No method, function, or main can be longer than 30 lines. Methods and functions must be cohesive and single-minded.

**Sample Input**: print teacher1 roster, remove a student from an empty roster, print teacher1 roster, add 3 students for teacher1, print teacher1 roster, remove one of the 3 student, print teacher1 roster, add the same student back again, print teacher1 roster, update a grade for the same student, print teacher1 roster, add 3 more students (3rd one can’t be added because roster is full and grades can’t be updated for this student who was not added), print teacher1 roster, add 3 student for teacher2, print teacher2 roster.

T1 P

T1 R Hedi

T1 P

T1 A Hedi G

T1 UQ Hedi 65

T1 UT Hedi 73

T1 UH Hedi 79

T1 UF Hedi 70

T1 A Joe V

T1 UQ Joe 75

T1 UT Joe 83

T1 UH Joe 99

T1 UF Joe 80

T1 A Fred M

T1 UQ Fred 93

T1 UT Fred 85

T1 UH Fred 91

T1 UF Fred 78

T1 P

T1 R Fred

T1 P

T1 A Fred M

T1 UQ Fred 93

T1 UT Fred 85

T1 UH Fred 91

T1 UF Fred 78

T1 P

T1 UH Fred 98

T1 P

T1 A Susan M

T1 UQ Susan 74

T1 UT Susan 85

T1 UH Susan 81

T1 UF Susan 75

T1 A Mia M

T1 UQ Mia 62

T1 UT Mia 73

T1 UH Mia 69

T1 UF Mia 65

T1 A Mark M

T1 UQ Mark 72

T1 UT Mark 73

T1 UH Mark 89

T1 UF Mark 76

T1 P

T2 A Sam M

T2 UQ Sam 64

T2 UT Sam 75

T2 UH Sam 69

T2 UF Sam 75

T2 A Mary G

T2 UQ Mary 82

T2 UT Mary 83

T2 UH Mary 89

T2 UF Mary 75

T2 A Sam M

T2 P

**Sample Output**:

T1: List of Students

T1: Can't remove student Hedi - Student doesn't exist

T1: List of Students

T1: Record added for Hedi

T1: Grade updated to 65 for Hedi

T1: Grade updated to 73 for Hedi

T1: Grade updated to 79 for Hedi

T1: Grade updated to 70 for Hedi

T1: Record added for Joe

T1: Grade updated to 75 for Joe

T1: Grade updated to 83 for Joe

T1: Grade updated to 99 for Joe

T1: Grade updated to 80 for Joe

T1: Record added for Fred

T1: Grade updated to 93 for Fred

T1: Grade updated to 85 for Fred

T1: Grade updated to 91 for Fred

T1: Grade updated to 78 for Fred

T1: List of Students

Hedi 65 73 79 70 72.25

Joe 75 83 99 80 84.20

Fred 93 85 91 78 86.30

T1: Record removed for Fred

T1: List of Students

Hedi 65 73 79 70 72.25

Joe 75 83 99 80 84.20

T1: Record added for Fred

T1: Grade updated to 93 for Fred

T1: Grade updated to 85 for Fred

T1: Grade updated to 91 for Fred

T1: Grade updated to 78 for Fred

T1: List of Students

Hedi 65 73 79 70 72.25

Joe 75 83 99 80 84.20

Fred 93 85 91 78 86.30

T1: Grade updated to 98 for Fred

T1: List of Students

Hedi 65 73 79 70 72.25

Joe 75 83 99 80 84.20

Fred 93 85 98 78 88.40

T1: Record added for Susan

T1: Grade updated to 74 for Susan

T1: Grade updated to 85 for Susan

T1: Grade updated to 81 for Susan

T1: Grade updated to 75 for Susan

T1: Record added for Mia

T1: Grade updated to 62 for Mia

T1: Grade updated to 73 for Mia

T1: Grade updated to 69 for Mia

T1: Grade updated to 65 for Mia

T1: Can't handle any more students

T1: No record found for student: Mark

T1: No record found for student: Mark

T1: No record found for student: Mark

T1: No record found for student: Mark

T1: List of Students

Hedi 65 73 79 70 72.25

Joe 75 83 99 80 84.20

Fred 93 85 98 78 88.40

Susan 74 85 81 75 78.60

Mia 62 73 69 65 67.20

T2: Record added for Sam

T2: Grade updated to 64 for Sam

T2: Grade updated to 75 for Sam

T2: Grade updated to 69 for Sam

T2: Grade updated to 75 for Sam

T2: Record added for Mary

T2: Grade updated to 82 for Mary

T2: Grade updated to 83 for Mary

T2: Grade updated to 89 for Mary

T2: Grade updated to 75 for Mary

T2: Record already exists for Sam

T2: List of Students

Sam 64 75 69 75 71.00

Mary 82 83 89 75 81.55

Normal Termination of Teacher Roster Program.