Object-Oriented Programming

Programming Assignment

## Submission Deadline: Friday, 27th June, 2014 (7:00 PM)

In this assignment, you are required to design a set of classes for storing student information, along with a main program that will read students’ information from a file, store the data, compute the total marks and grades, and then writes a summary report to an output file. There should be one base class (**Student**) to store common data, and three derived classes that divide the set of students into three categories: English students (**EngStudent**), Computer Science students (**CompStudent**), and Math students (**MathStudent**). All the data members of these classes should be declared as *protected*. The **Student** class will have the following data members:

* Student’s roll number (an integer)
* Student’s name (a string, use STL’s string type **#include <string>**)
* Total marks of the student (double)
* Grade of the student (char)

Each derived class will have appropriate data members to store the marks of different assessments applicable for a particular student. You can assume that the marks of each assessment are given out of 100.

Assessments and their weightages for the three categories of students are as follows:

* **English** – Attendance = 10%, Report = 30%, Midterm = 30%, Final Exam = 30%
* **Computer Science** – Project = 25%, Midterm = 35%, Final Exam = 40%
* **Math** – Quiz Average\* = 25%, Homeworks = 25%, Final Exam = 50%

\**There are a total of 5 quizzes, to be averaged into one Quiz Average (which can be a decimal number).*

Note that the constructors of these four classes will take only two parameters (student’s roll number and name). Total marks and grade will be calculated later on.

The base class (Student) will be an ***abstract*** class. It will have these three ***pure virtual functions***. Each of the three derived classes will implement these functions differently:

* void readMarks (ifstream& fin) for reading the marks of different assessments for a particular student from the file handle given as argument.
* void calculateMarks () for calculating the total marks (out of 100) and grade of a particular student.
* void writeRecord (ofstream& fout) for writing the marks of different assessments, total marks, and grade of a particular student in the file opened by the file handle.

The flow of your program will be as follows:

1. Ask the user for input and output file names. This is the only input and output that should be done from keyboard and to the screen. All other input and output will be done from the files. See the sample execution given below (text shown in red is entered by the user):

|  |
| --- |
| **Enter the name of input file: input.txt**  **7 Student Records have been read.**  **Enter the name of output file: output.txt**  **Processing complete. See “output.txt” for students’ grade summary.** |

1. After reading the name of input and output files, your program should read the student data from the input file and store it in an array. You should use just one array for all students, not a separate array for each subject (i.e. **this should be an array of** Student **pointers**). You will need to allocate this array dynamically, since the size (No. of students) is stored in the input file. Each student’s data should be stored in a separate object of the appropriate type (EngStudent, CompStudent, or MathStudent), which will also be allocated dynamically through an index of the Student pointers array. A sample input file (input.txt) is shown below:

|  |
| --- |
| **7**  **221 Sherlock Holmes**  **MT 60 72 85 65 62 75 80**  **567 Dr. Watson**  **CS 70 85 80**  **1234 Robert Langdon**  **EN 85 91 79 93**  **123 Bill Gates**  **CS 65 75 81**  **808 Albert Einstein**  **MT 45 79 71 65 91 72 78**  **665 Donald Knuth**  **EN 92 97 85 90**  **778 Alan Turing**  **CS 93 88 95** |

The first line of the input file tells the number of student records present in the file. Each student record takes two lines. The first line contains the roll number and name of the student. In the second line, the first two characters specify the type of the student (EN for English student, CS for Computer science student, and MT for Math student). After that the marks of that student for different assessments are given (separated by spaces). Note once again that the assessments are different for each type of student (as described on Page 1).

1. After reading the records of students, your program should calculate the total marks and grade for each student. Then, it should generate a summary file of student grades. The output file (output.txt) generated from the input file (input.txt) shown on previous page will be as follows:

|  |
| --- |
| **Students’ Grade Summary**  **=======================**  **Total Students: 7**  **ENGLISH CLASS**  **=============**  **Attendance (A, 10%), Project (P, 30%), Midterm (M, 30%), Final (F, 30%)**  **Roll No. Student Name A P M F Total Grade**  **-------------------------------------------------------------------**  **1234 Robert Langdon 85 91 79 93 87.4 B**  **665 Donald Knuth 92 97 85 90 90.8 A**  **COMPUTER SCIENCE CLASS**  **======================**  **Project (P, 25%), Midterm (M, 35%), Final (F, 40%)**  **Roll No. Student Name P M F Total Grade**  **--------------------------------------------------------------**  **567 Dr. Watson 70 85 80 79.25 C**  **123 Bill Gates 65 75 81 74.9 C**  **778 Alan Turing 93 88 95 92.05 A**  **MATH CLASS**  **==========**  **Quiz Average (Q, 25%), Homeworks (H, 25%), Final (F, 50%)**  **Roll No. Student Name Q H F Total Grade**  **-----------------------------------------------------------------------**  **221 Sherlock Holmes 68.8 75 80 75.95 C**  **808 Albert Einstein 70.2 72 78 74.55 C** |

|  |  |
| --- | --- |
| **Marks Range** | **Grade** |
| >= 90 | A |
| 80 – 89 | B |
| 70 – 79 | C |
| 60 – 69 | D |
| < 60 | F |

Note that the student records in the output file are displayed according to their subjects. In order to accomplish this, you will need to traverse your array three times (firstly, to display all English students, then for Computer science students, and finally for Math students).

Use the table shown on right to assign grades to students:

**Important Points**

* You must use inheritance and polymorphism (as described in the assignment statement) to implement the class hierarchy and the afore-mentioned functionalities.
* You can assume that the input file is correct according to the format given above.
* The output file generated by your program should exactly follow the format given above.

These *good programming practices* will also have their (significant) weightage in the marking of this assignment:

* There should be no memory leaks, dangling pointers, etc. in your program.
* Comment your code intelligently. (Uncommented code may not be given any credit)
* Use meaningful variable and function names.
* Indent your code properly.
* Do not use any global variables.

Moreover, if your submitted program gives an error or a warning message at the time of compilation, you will get a ZERO in the assignment.

**Instructions**

* You are required to do this assignment on your own. Absolutely NO collaboration is allowed. Any traces of plagiarism would result in an “F” grade in this course.
* Read the complete assignment statement carefully before starting the implementation. Make sure that your implementation adheres to all the requirements mentioned for each function.
* Submit your assignment in the folder specified at the start. No submissions will be accepted through email.
* *Late Submission:* You can late-submit this assignment according to the following rules. **Note that if you submit more than one copy of your assignment, only the latest submission will be marked. All previous submissions will be deleted.**
  + Submission within 24 hours of the deadline will result in 25% deduction
  + Submission within 48 hours of the deadline will result in 50% deduction
  + Submission after 48 hours of the deadline will not be considered

**☺ GOOD LUCK! ☺**