CSCI204/MCS9204 Assignment 1

(Total 10 marks, Due by 11:59 pm sharp on Sunday, 30 August, 2015)

Aims

This assignment aims to establish a basic familiarity with C++ classes. The assignment introduce increasingly object-based, C++ style of solution to a problem.

General Requirements

- You should observe the common principles of OO programming when you design your classes.
- You should make proper documentation and implementation comments in your codes where they are necessary.
- Logical structures and statements are properly used for specific purposes.

Objectives

On completion of these tasks you should be able to:

- code and run C++ programs using the development environment.
- make effective use of the on-line documentation system that supports the development environment.
- code programs using C++ in a hybrid style (procedural code using instances of simple classes) and in a more object-based style.
- manipulate string data.

Tasks:

Task 1: (6 marks)

Write a program to read a list of students from a data file, the name of which should be input from a command line, and create a dynamic array to store the students’ records. Do not use fixed size array to store the student records.

The format of the data file is as the following:

```
Albert Einstein 52 -1 67 63
Steve Abrew 90 86 90 93
David Nagasake 100 85 93 89
Mike Black 81 87 81 85
Andrew Van Den 90 82 95 87
Joanne Dong Nguyen 84 80 95 91
Chris Walljasper 86 100 96 89
Fred Albert -1 70 68 -1
Dennis Dudley 74 79 77 81
Leo Rice -1 -1 95 -1
Fred Flinstone 73 81 78 74
Frances Dupre 82 -1 76 79
Dave Light 89 76 91 83
Hua Tran Du 91 81 87 94
Sarah Trapp 83 -1 98 -1
```

Each line shows one student’s record with the name and marks for different subjects. The mark value -1 means the subject hasn’t been enrolled by the student.

Define a class **Student** in a file **student.h** according to the description below:

- The class has a private data member **name** as a **char pointer**;
- The class has a private data member **marks** that store marks of all subjects (up to 4 subjects);
- The class has a private data member **num** that holds the number of subjects that have been enrolled by the student.
- Define appropriate constructors and a destructor.
- Define necessary public member functions that can set and get values of data members.
- Define a public print function that can print out a student’s record with formatted output (see the Testing of this task for more details).

Implement member functions of Student in a file **student.cpp**.

Implement main function and other functions in a file **task1main.cpp** to ask a user input a text file name; count how many records stored in the text file and create dynamic memory for students;   load records from the given text file into the dynamic array.

After all records have been loaded, the program is to print the subjects’ marks for each student along with the total marks and average marks. After all records have been printed, the program is to print the total number of enrolled students, mean and standard deviation for each subject by using formatted output. Check the section Testing for more details. You may need to do some research to understand how to compute mean and standard deviation.

Note: The subject that students haven't enrolled in cannot be count in for computing total, average marks, mean and standard deviation of the subject.

You need create dynamic memory to store student's name for this task. Make sure there is no memory leak.

Testing:

Use g++ to compile the source files by
```
g++ –o task1 task1main.cpp student.cpp
```
and run the program (red data means input by a user).

```
./task1
Input a file name: students.dat
          Name              mark_1  mark_2  mark_3  mark_4  Total Average
************************************************************************
Albert Einstein                 52              67      63    182   60.67
Steve Abrew                     90      86      90      93    359   89.75
David Nagasake                 100      85      93      89    367   91.75
Mike Black                      81      87      81      85    334   83.50
Andrew Van Den                  90      82      95      87    354   88.50
Joanne Dong Nguyen              84      80      95      91    350   87.50
Chris Walljasper                86     100      96      89    371   92.75
Fred Albert                             70      68            138   69.00
Dennis Dudley                   74      79      77      81    311   77.75
Leo Rice                                        95             95   95.00
Fred Flinstone                  73      81      78      74    306   76.50
Frances Dupre                   82              76      79    237   79.00
Dave Light                      89      76      91      83    339   84.75
Hua Tran Du                     91      81      87      94    353   88.25
Sarah Trapp                     83              98            181   90.50
************************************************************************
Number of students              13      11      15      12
Mean                         82.69   82.45   85.80   84.00
Standard Deviation           11.24    7.18   10.11    8.50
************************************************************************
```
The input testing file **students.dat** can be downloaded from the subject Moddle site.

Note: Your program should work on different testing data files with the same format.

To check the memory leaks you can run the program:
```
becheck ./task1
```

Task2: (4 marks)

Define a class **Point** in a header file **shapes.h** that contains:

- Data members of x and y coordinates;
- Initialization and copy constructors;
- Member functions that can set new values for the coordinates x and y;
- Member functions that return x and y coordinates from a Point object.

Implement member functions of the class Point in a file **shapes.cpp**.

Define a class **Line** in the header file **shapes.h** that contains:

- Data member of two points, which can identify a line;
- Initialization and copy constructors;
- Member functions that can set new values for those data members;
- Member functions that can get those points from a Line object.

Implement member functions of the class Line in the file **shapes.cpp**.

Define a class Circle in the header file shapes.h that contains:

- Data member of centre point and radius, which can identify a circle;
- Initialization and copy constructors;
- Member functions that can set new values for centre and radius;
- Member functions that can get the centre points and radius from a Circle object.

Implement member functions of the class Circle in the file **shapes.cpp**.

Write a main() function in a file **task2main.cpp** to test those constructors and member functions that described above. For a given Circle and a Line objects, check if they are intersected?

Hint:
Assume a line is defined by two points (x1, y1) and (x2, y2). A point is located at (x0, y0). The program may use the following expression to calculate the distance from the position of a point to the line:

$$distance = |(x2-x1)(y1-y0)-(x1-x0)(y2-y1)| / sqrt((x2-x1)^2 + (y2-y1)^2)$$

Do not forget namespace in your program files.

Testing:

Use g++ to compile the source files by

```
g++ –o task2 task2main.cpp shapes.cpp
```

and run the program by

```
./task2
```

The output of the program should be looked like the following:

```
Create a point p1(10, 20)
Copy a point p2(10, 20)
Set new coordinates for p2(100, 200)
Create a line (10, 20) - (100, 200)
Change point of the line (30, 40) - (100, 200)
Create a circle (80, 10), radius=50
Copy a circle (80, 10), radius=50
A line (30, 40) - (100, 200) and a circle(80, 10) of radius 50 are not intersected.
distance=57
```

You can test the program by using different values.

Submission

This assignment is due by 11.59 pm (sharp) on Sunday August 30, 2015.

Assignments are submitted electronically via the ***submit*** system.

For this assignment you must submit the files **task1main.cpp, student.h, student.cpp, task2main.cpp, shapes.h and shapes.cpp** via the command:

```
$ submit -u your_user_name -c CSCI204 -a 1 task1main.cpp student.h student.cpp
task2main.cpp shapes.h shapes.cpp
```

and input your password.

Make sure that you use the correct file names. The Unix system is case sensitive. You must submit all files in one ***submit*** command line.

Your program code must be in a good programming style, such as good names for variables, methods, classes, and keep indentation.

Submission via e-mail is NOT acceptable.

After submit your assignment successfully, please check your email of confirmation. **You would loss 50% ~ 100% of the marks if your program codes could not be compiled correctly.**

Late submissions do not have to be requested. Late submissions will be allowed for a few days after close of scheduled submission (up to 3 days). Late submissions attract a mark penalty; this penalty may be waived if an appropriate request for special consideration (for medical or similar problem) is made via the university SOLS system *before* the close of the late submission time. No work can be submitted after the late submission time.

A policy regarding late submissions is included in the course outline.

The assignment is an **individual assignment** and it is expected that all its tasks will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc. please consult your lecturer or tutor during tutorial classes or office hours. Plagiarism will result in a **FAIL** grade being recorded for that assessment task.

End of specification