Assignment 4

Assignment Instructions:

- Complete the following programming exercises in C/C++.
- Aim to make your code as concise and logical as possible.
- For convenience, you can write all functions and classes in a single file.
- Write functions to be self-contained units of code, i.e. the code inside your functions must **not** rely on variables you define outside the function (i.e. no *global variables*).
- Classes can be written as one block (i.e. there is no need to separate method declarations from their definitions).
- It's recommended that you test your code by calling your functions from your main() method, but no code written in main() will be graded.
- Do not use any library functions, unless the question specifies that you can.
- Submit your answers as a ***single text .TXT document***, *no zip or Word files etc.*, with **each answer clearly numbered and your code properly formatted/indented** (use Ctrl+A then Ctrl+I in Code::Blocks to properly format your code after you are finished writing it).

This assignment is due 6 pm Monday 9th March. No late submissions accepted.

Q1. Write a **Line** class to represent a line in a 2D plane. The class has two data members defining a line, a **slope** and an **intercept**. It must not have any other data members. It should have the following public methods:

- A **constructor** taking a slope and an intercept value to make a new line
- A **constructor** that takes a set of point coordinate values x1, y1, x2, y2, which define a line passing through the points (x1,y1) and (x2,y2)
- A method **double distanceTo(Point p)** which returns the distance between the line and a given point object **p**
- A method **Point intersection(Line ln)** which returns the intersection point of this line and a given line **ln**.

Use the **Point** class given below for the Point type needed in the last two methods (put it in your code before your Line class). Do not modify this Point class.

```
class Point {
private:
    double x, y;
public:
    Point(double _x, double _y) : x(_x), y(_y) {}
    double _x() { return x; }
    double _y() { return y; }
    double distanceTo(const Point & p) { return sqrt((x-p.x)*(x-p.x) + (y-p.y)*(y-p.y)); }
};
```

Note: you may use the **fabs()** and **sqrt()** methods from the standard maths library.

[15 marks]

Q2. Write a function (not a method of a class), that uses the functionality already provided by your Line class, to solve any pair of simultaneous equations of the form:

$a_1 x + b_1 y + c_1 = 0$

$a_2 x + b_2 y + c_2 = 0$

Your function should have the following form:

```
Point solve(double a1, double b1, double c1, double a2, double b2, double c2) { /* use Line objects here to find the answer - don't rewrite any code unnecessarily */ }
```

Test your method using the following code in main() :

```
 Point sol = solve(2.5, -2.0, 3.0, 1.5, -3.0, -4.1);
 cout << "Equations solution is x = " << sol._x() << " and y = " << sol._y() << endl;
```

which should print out

```
Equations solution is x = -3.82222 and y = -3.27778
```

Note: Aim to have no more than 3 or 4 lines of code in your function.

[15 marks]

Note: In the next three questions you will create a set of classes that are related by inheritance. It is recommended that you read all three questions before writing any code. Sketching out a class diagram as you read the questions will help you to get a picture of the required structure of the classes (you do not need to submit this diagram).

Q3. Write an *abstract* class to represent a **Student**. A student has an **ID**, a **first name** and a **last name**. The ID should be an integer and the first and last names should be C++ strings. The class must have the following methods:

- A **constructor** that takes a student ID value, a first name and a last name
- A **int getID()** method which returns the student's ID value
- A pure virtual method **string getType()** which, if implemented in any class derived from Student, would be expected to return the type of student as a string
- A virtual **void printInfo()** method which prints out the student's details. This method should call the **getType()** method to print out the student type as the first line of the printout.

[10 marks]

Q4. Write a *concrete* class called **ResearchStudent** which inherits from your Student class of question 3. A research student has a supervisor name (as a single C++ string) and a thesis title (as a C++ string). The class must also have the following methods:

- A **constructor** taking the student's ID, first and last names, supervisor name and thesis title. This constructor calls the parent class constructor.
- A **void printInfo()** method that calls the parent class's printInfo() method to print generic student information and also prints details specific to a research student.
- A method that provides an implementation (overrides) the base class pure virtual function **getType()**. This method should simply return the string "Research Student".

[15 marks]

Q5. Write a concrete class called **TaughtStudent** which also inherits from your Student class. A taught student has an array of module names for which they are registered. This array can have up to 10 module names. A module name is a C++ string. Additionally, the class has a member **numRegisteredModules** which stores the number of modules for which the student is currently registered. The class must have the following methods:

- A **constructor** taking the student's ID, first and last name. The constructor should set numRegisteredModules to 0.
- A method **bool registerFor(const string & moduleName)** which adds a module to the student's modules. If the student is already registered for 10 modules then this method should return false, otherwise it returns true.
- A **void printInfo()** method that calls the parent class printInfo() method and also prints details specific to a taught student, including their list of registered modules.
- An override for the base class pure virtual function that returns the string "Taught Student".

[15 marks]

Q6. Write a function (not a method of a class) **void studentTester()** which contains code to test the classes you have developed in the previous three questions. When this function is called it should:

- Create two research students.
- Create two taught students, and register them for three modules each.
- Make an **array of pointers to Student** and point the elements of the array at your four student objects.
- Run a for loop which calls each student's printInfo() method, using the array of pointers.

[15 marks]

Q7. Write a class **Employee**. An employee has a number and a name and the following additional features:

- The first employee created is automatically assigned the number 1001, the second 1002 and so on. Implement this feature using a static data member **nextEmpolyeeNumber**.
- Also keep track of the total number of employee objects using another static data member **totalNumberEmployees**. Note that this number should be decremented when an Employee object is destroyed.
- Add a static member function called **numEmpolyees()** which returns the current number of employee objects.

Using the following code in main(), explain why this code "leaks memory".