CSCI204/MCS9204 Assignment 2

(Total 15 marks, Due by 11:59 pm sharp on Sunday, 27 September, 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: Overloading operators (8 marks)

A polynomial has the form
$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_n x^n$.

In C++, we can store a polynomial in an array. For example:
An array f[5] that contains coefficients: {3.7, 5.2, 12.4, 0, 8.0} represents a polynomial

$f(x) = 3.7 + 5.2x + 12.4x^2 + 8.0x^4$.

The degree of term 1 of above polynomial f(x) is zero, coefficient is 3.7. The degree of term 2 of above f(x) is 1, coefficient is 5.2. The degree of term 3 of above f(x) is 2, its coefficient is 12.4, etc.. The degree of above polynomial is 4 (highest degree of all terms).
There are some operations defined for polynomials. Assume there are two polynomials f(x) and g(x) defined as following:
$f(x) = 5x^2 + 4x^3 + 2x^4$,
$g(x) = 1 + 5x^2 + 23x^3 + 7x^4 + 15x^5 + 9x^6$.

We define **addition** of two polynomials as additions of coefficients of terms with the same degrees in two polynomials. For example:
$f(x) + g(x) = 1 + 10x^2 + 27x^3 + 9x^4 + 15x^5 + 9x^6$.

We define **subtraction** of two polynomials as subtraction of coefficients of terms with the same degrees in two polynomials. For example:
$f(x) - g(x) = -1 - 19x^3 - 5x^4 - 15x^5 - 9x^6$,

We define **multiplication** of two polynomials as multiply each term in one polynomial with each term in another polynomial (multiply coefficients, add degrees), then add all of them together. For example:

$$\begin{aligned} f(x) * g(x) &= (5x^2 + 4x^3 + 2x^4) * (1 + 5x^2 + 23x^3 + 7x^4 + 15x^5 + 9x^6) \\ &= 5x^2 * (1 + 5x^2 + 23x^3 + 7x^4 + 15x^5 + 9x^6) + \\ &\quad 4x^3 * (1 + 5x^2 + 23x^3 + 7x^4 + 15x^5 + 9x^6) + \\ &\quad 2x^4 * (1 + 5x^2 + 23x^3 + 7x^4 + 15x^5 + 9x^6) \\ &= 5x^2 + 4x^3 + 27x^4 + 135x^5 + 137x^6 + 149x^7 + 119x^8 + 66x^9 + 18x^{10} \end{aligned}$$

We define **division** and **remainder** of polynomials as: if f(x) = g(x) * d(x) + r(x), which degree of r(x) is lower than g(x), then f(x) / g(x) = d(x), f(x) % g(x) = r(x). For example:
$5x^2 + 4x^3 + 2x^4 = (1 + 5x^2 + 23x^3 + 7x^4 + 15x^5 + 9x^6) * 0 + (5x^2 + 4x^3 + 2x^4)$.
So
f(x) / g(x) = 0, and f(x) % g(x) = f(x).

$1 + 5x^2 + 23x^3 + 7x^4 + 15x^5 + 9x^6 = (5x^2 + 4x^3 + 2x^4) * (-4.75 - 1.5x + 4.5x^2) + (1 + 28.75x^2 + 49.5x^3)$.
So
$g(x) / f(x) = -4.75 - 1.5x + 4.5x^2$,
$g(x) \% f(x) = 1 + 28.75x^2 + 49.5x^3$,

Define a class **Polynomial** in a file **polynomial.h**, implement member functions and friend functions in a file **polynomial.cpp**. In the class Polynomial, you should define data members, friends and function members that specified below:

- A data member that can be used to store the length of an array for a polynomial. The length of the array will be the degree of the polynomial plus one.
- A pointer of double that can be used to store coefficients of a polynomial;
- Extraction operator (>>) can read a polynomial from a given input stream. First it will read an integer of the length of a polynomial, then read coefficients of the polynomial. To input a polynomial, such as f(x) = $5 x^2 + 4 x^3 + 2 x^4$, the input data will be:
  5
  0 0 5 4 2
- Insertion operator (<<) can display a polynomial;
- Assignment operator (=) can make deep copy of a polynomial object;
- Addition operator (+) can add two polynomials and return the result;
- Subtraction operator (-) can subtract a polynomial from another polynomial;
- Multiplication operator (*) can compute multiplication of two polynomials;
- Division operator (/) can compute division of two polynomials;
- Remainder operator (%) can compute remainder when one polynomial divided by another polynomial.
- You may add other necessary constructors, a destructor, and other member functions.

Implement C++ driver program include main function in a file **testPolynomial.cpp** to test all above functions and operators.
When the program starts, it will ask a user to input the first polynomial, and the second polynomial from keyboard. Then the program will test copy constructor, overloading operators as following:

Input polynomial p1:
5
0 0 5 4 2
p1(x) = 5 x^2 + 4 x^3 + 2 x^4

Input polynomial p2:
7
1 0 5 23 7 15 9
p2(x) = 1 + 5 x^2 + 23 x^3 + 7 x^4 + 15 x^5 + 9 x^6

Copy p2 to p3, p3(x) = 1 + 5 x^2 + 23 x^3 + 7 x^4 + 15 x^5 + 9 x^6

p3(x)=p1(x)+p2(x) = 1 + 10 x^2 + 27 x^3 + 9 x^4 + 15 x^5 + 9 x^6

p3(x)=p1(x)-p2(x) = -1 - 19 x^3 - 5 x^4 - 15 x^5 - 9 x^6

p3(x)=p2(x)-p1(x) = 1 + 19 x^3 + 5 x^4 + 15 x^5 + 9 x^6

p3(x)=p1(x) * p2(x) = 5 x^2 + 4 x^3 + 27 x^4 + 135 x^5 + 137 x^6 + 149 x^7 + 119 x^8 + 66 x^9 + 18 x^10

p3(x) = p2(x) / p1(x) = -4.75 - 1.5 x + 4.5 x^2

p3(x) = p1(x) / p2(x) = 0

p3(x) = p1(x) % p2(x) = 5 x^2 + 4 x^3 + 2 x^4

p3(x) = p2(x) % p1(x) = 1 + 28.75 x^2 + 49.5 x^3

Note: numbers in red are the inputs from keyboard. To display degree of a term, you can use x^i to represent x^i.

Testing:

Use CC to compile the source files on banshee by
g++ –o task1 testPolynomial.cpp Polynomial.cpp

You can test the task by using the polynomials defined in a text file **input1.txt** (same as the polynomial above) by
$ task1 < input1.txt

To check memory leak, you may run the program by
$ bcheck task1 < input1.txt

To test the program by using polynomials defined in a file **input2.txt**, you can run the program by
$ task1 < input2.txt
Input polynomial 1: p1(x) = 1 + 2 x + x^2 + x^3

Input polynomial 2: p2(x) = 1 + 2 x + x^2 + 2 x^3 + 3 x^4

copy p2 to p3. p3(x) = 1 + 2 x + x^2 + 2 x^3 + 3 x^4
p3(x)=p1(x) + p2(x) = 2 + 4 x + 2 x^2 + 3 x^3 + 3 x^4

p3(x)=p1(x) - p2(x) = - x^3 - 3 x^4

p3(x)=p2(x) - p1(x) = x^3 + 3 x^4

p3(x)=p1(x) * p2(x) = 1 + 4 x + 6 x^2 + 7 x^3 + 10 x^4 + 9 x^5 + 5 x^6 + 3 x^7

p3(x) = p2(x) / p1(x) = -1 + 3 x

p3(x) = p1(x) / p2(x) = 0

p3(x) = p1(x) % p2(x) = 1 + 2 x + x^2 + x^3

p3(x) = p2(x) % p1(x) = 2 + x - 4 x^2

The input testing files **input1.txt** and **input2.txt** can be downloaded from the elearning space.

Note: Your program should work on different testing data files. There should be NO memory leak.

Task2: Class inheritance (7 marks)

The base class Vehicle is defined in a file **vehicle.h**. Download the file **vehicle.h** from Moodle site. You can use the base class to define derived classes Car in **car.h**, Bus in **bus.h** and Truck in **truck.h**.

The derived class Car consist extra data members:

- model: with 15 characters length to record the model, such as “Cruze”, “Apollo”;
- size: Can be Small, Mid, Sport, Luxury;
- type: Can be Manual or Automatic;
- number_of_seats: Total number of seats.

The derived class Bus consist extra data members:

- model: with 15 characters length to record the model;
- type: Can be Mini, Normal, Double, School;
- number_of_seats: Total number of seats.

The derived class Truck’s consist extra data members:

- model: with 15 characters length to record the model;
- type: Can be Light, Formal, Heavy;
- gross: Gross weight the truck can carry, unit is kg.

Define necessary constructors and member functions for the base class and derived classes specified above in there header files. Define overloading extraction (>>) and insertion (<<) operators for the classes Vehicle, Car, Bus and Truck that read data from a text file or save data into a text file. Implement those member functions for the classes Vehicle, Car, Bus and Truck in the files **vehicle.cpp**, **car.cpp**, **bus.cpp** and **truck.cpp** accordingly.

Define a class **VehicleManagement** in a file **vehicleManagement.h** contains two data members: A pointer array of Vehicle (where each element is a pointer of the base class Vehicle) and the size of the array. Define the following member functions in the class VehicleManagement:

- A function loadVehicles(ifstream &) that allocate dynamic memory for the pointer array of Vehicle, loads vehicles from the text file that has been given. Display the number of vehicles that have been loaded.

- A function addVehicle()that get a plate number, check is the vehicle already exists or not. If the plate number does not exist, the function will get new vehicle information and add it into the vehicle array.
- A function displayVehicle() that gets input plate number, finds the vehicle and displays the vehicle information.
- A function renewVehicle() that gets input plate number, finds the vehicle and renews the registration.
- A function transformVehicle() that gets input plate number, finds the vehicle and transforms it to a new owner.
- A function deleteVehicle() that gets input plate number, finds the vehicle, displays the vehicle information, and deletes it when confirmed.
- Other member necessary functions. Such as run(), which display menu and get choice, call the correspondent member functions until "Quit" has been selected.

Implement the member functions of the class VehicleManagement in a file **vehicleManagement.cpp**.

Implement main() and other necessary functions in a file **task2main.cpp** to create instance of VehicleManagement, call the member function run() starts to simulate the process of vehicle registration management.

When the program starts, it loads vehicles registration records from a given text file into a dynamic array of Vehicle pointers (which is a base class pointer array).

Input file name: vehicles.dat
12 records loaded

Each line of the text file stored one record of a vehicle. The first character of a line indicates the registered vehicle is a car (c), or a bus (b) or a truck (t). The format of the text file should like follows:

t, one ton, Light, 1000.0, AAA-100, Toyota, 10/02/2002, 03/05/2002, 1034.0, Roadwalker, 10 Smith street Wollongong NSW 2500, 12345678
c, gs105, Small, Manual, 4, ABC-110, Ford, 13/12/2002, 04/01/2003, 985.0, Butcher, 23 Hole ave. Norwa NSW 2534, 12246753
b, tds213, Mini, 12, DUV-321, Benz, 05/01/2003, 20/04/2003, 2285.0, Cooker, 18 Mountain street West Wollongong, 21345786

You can download the text file vehicles.dat from the Moodle site.

Then the program displays a menu, get input choices and call the correspondent functions until "Quit" has been chosen.

1. Register new vehicle
2. Display vehicle registration
3. Renew vehicle registration
4. Transform vehicle registration
5. Delete vehicle registration

6. Quit
Choice: 1

- If the user chooses 1 (register a new vehicle), the program will ask the user input information for a new car/bus/truck registration as following

Plate number: ABC-001

The letters in plate number should be upper cases. If the user input lower cases letters for plate number, the program should transform them into upper cases. After the user type the plate number, the program should check if the plate number has already been used. If answer is yes, the program should prompt error message and ask the user type another plate number until the plate number is not exist. Then the program ask the user input more information like follows

Choose vehicle (1-car, 2-bus, 3-truck): 1
Made by: Benz
Produced: 02/10/2010
Register date: 16/05/2011
Gross weight: 1052.0
Owner name: Sailor
Corresponding address: 123 Super market street Fairy Meadow NSW 2510
Contact phone: 11223344

The register date can be got from the current date from the computer by using time(). The rest information should be input according to the vehicle type.
If the vehicle is a car, the program will get following information:

Model: rts110
Size (1-small, 2-middle, 3-sport, 4-luxury): 3
Type (1-manual, 2-auto): 2
Number of seats: 5

If the vehicle is a bus, the program will get input

Model: Camry
Type (1- mini, 2-normal, 3-double, 4-school): 2
Number of seats: 35

If the vehicle is a truck, the program will get input

Model: Dgs2002
Type (1- light, 2-formal, 3-heavy): 2
Can carry (KGs): 30000.0

Then add the new vehicle in the container object.

- If the user chooses 2 (display vehicle registration), the program gets input

Plate number: ABC-001

Then the program searches the vehicle from the array of vehicles. If the vehicle exists, then the program displays its information.

- If the user chooses 3 (renew vehicle registration), the program gets input of plate umber

Plate number: ABC-001
Then the program searches the vehicle from the dynamic array. If the vehicle exists, then the program display its information and ask the user confirm the renew option
Renew registration? (Y/N) Y
If it is Y (or y) the program sets the registedDate of the vehicle with the current date of the computer and updates the record in the container object.

- If the user chooses 4 (transform vehicle registration), the program gets input

Plate number: ABC-001
Then the program searches the vehicle from the dynamic array. If the vehicle exists, then the program displays its information and asks the user input a new owner's information
Owner name: David
Corresponding address: 21 Cave street Albin Park NSW 2520
Contact phone: 77882211
Then update the record in the container object.

- If the user chooses 5 (Delete vehicle registration), the program gets input

Plate number: EDU-012
Then the program searches the vehicle from the dynamic array. If the vehicle exists, then the program displays its information and asks the user to confirm the deletion
Are you sure? (Y/N) Y
Then the record in the container object should be removed.

- If the user chooses 6 (Quit), the program will get file name and save all records in a given text file, then quit.

Text file name: myvehicles.dat
15 car records have been saved

Testing:

Use g++ to compile the source files by
g++ –o task2 task2main.cpp vehicle.cpp car.cpp bus.cpp truck.cpp vehicleManagement.cpp
and run the program by
./task2

The output of the task should look like the output above.

You can test the program by using different values.

Submission

This assignment is due by 11.59 pm (sharp) on Sunday September 27, 2015.

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

For this assignment you must submit all the files via the command:

$ submit -u your_user_name -c CSCI204 -a 2 polynomial.h polynomial.cpp testPolynomial.cpp task2main.cpp vehicle.h vehicle.cpp car.h car.cpp bus.h bus.cpp truck.h truck.cpp vehicleManagement.h vehicleManagement.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