CSCI804 Assignment 2

(Total 10 marks, Due by 11:59 pm sharp on Sunday, 17 April, 2016)

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.
- Familiar with overloading operators.
- Understand class inheritance.

Tasks:

Task 1: (4 marks)

A complex number has the form
$a + b\,i$,

where *a*, *b* are real numbers, *i* is the imaginary unit, $i^2 = -1$. It is very useful in mathematics, physics, engineering, economics.

In this task, you will define a class **ComplexNumber** in a file **ComplexNumber.h** and implement the C++ program code in a file **ComplexNumber.cpp**.
In the class **ComplexNumber**, declare data members to store a complex number. Define the following functions in the class ComplexNumber.

- In the class ComplexNumber, you will define default constructor, copy constructor and other necessary constructors;
- In the class ComplexNumber, define following overloading operators:
    - Extraction operator (>>) to get input values from keyboard;
    - Insertion operator (<<) to print out the a complex number;
    - Addition operator (+) to compute two complex numbers' addition. The addition of two complex numbers is defined as:
      $c_1 + c_2 i = (a_1 + a_2 i) + (b_1 + b_2 i)$,
      where $c_1 = a_1+b_1$, $c_2 = a_2 + b_2$.
    - Addition assignment operator (+=) to compute two complex numbers' addition.
    - Subtraction operation (-) to compute two complex numbers' subtraction. The subtraction of two complex numbers is defined as:
      $c_1 + c_2 i =(a_1 + a_2 i) - (b_1 + b_2 i)$,
      where $c_1 = a_1 - b_1$, $c_2 = a_2 - b_2$.
    - Subtraction assignment operator (-=) to compute two complex numbers' subtraction.
    - Multiplication operator (*) to compute two complex numbers' multiplication. The multiplication of two complex numbers is defined as:
      $c_1 + c_2 i =(a_1 + a_2 i) * (b_1 + b_2 i)$,
      where $c_1 = a_1 * b_1 - a_2 * b_2$, $c_2 = a_1 * b_2 + a_2 * b_1$.
    - Multiplication assignment operator (*=) to compute two complex numbers' multiplication.
    - Multiplication operator (*) to compute a complex number times a double value. The multiplication of a complex number is defined as:
      $c_1 + c_2 i =(a_1 + a_2 i) * b$,
      where $c_1 = a_1 * b$, $c_2 = a_2 * b$.
    - Multiplication operator (*) to compute a double value times a complex number. The multiplication of a double value times a complex number is defined as:
      $c_1 + c_2 i =b * (a_1 + a_2 i)$,
      where $c_1 = b * a_1$, $c_2 = b * a_2$.
    - Division operator (/) to compute two complex numbers' division. The division of two complex numbers is defined as:
      $c_1 + c_2 i = (a_1 + a_2 i) / (b_1 + b_2 i)$
      $= ((a_1 + a_2 i) / (b_1 - b_2 i))/ ((b_1 + b_2 i) * (b_1 - b_2 i))$,
      where $c_1 = (a_1*b_1 + a_2 * b_2) / (b_1^2+b_2^2)$, $c_2 = (a_2 * b_1 – a_1 * b_2) / (b_1^2+b_2^2)$.
      The function should throw a string exception if the divisor is zero.

- Division assignment operator (/=) to compute two complex numbers' division.
  The function should throw a string exception if the divisor is zero.

Implement **main()** in a file **task1Main.cpp** to test the functions and operators that defined above (See the Testing examples of the task for more details). **In the main() function, exceptions should be caught and message should be displayed.**

Testing:

Use g++ to compile the source files on banshee by

```
$ g++ –o task1 task1Main.cpp ComplexNumber.cpp
```

You can test the task by using the data defined in a text file **input1.txt** by

```
$ ./task1
```

and input data that required. Your program will print out results like following (Red data means input from keyboard):

```
Input a complex number for cn1 :  10  4
Complex number cn1 = 10+4i
Input a complex number for cn2 :  5  2
Complex number cn2 = 5+2i
Complex number cn3 = cn1 + cn2 = 15+6i
Complex number cn3 = cn1 = 10+4i
Complex number cn3 += cn2 = 15+6i
Complex number cn3 = cn1 - cn2 = 5+2i
Complex number cn3 = cn1 = 10+4i
Complex number cn3 -= cn2 = 5+2i
Complex number cn3 = cn1 * cn2 = 42+40i
Complex number cn3 = cn1 = 10+4i
Complex number cn3 *= cn2 = 42+40i
Input a real number: 3.4
Complex number cn3 = cn1 * 3.4 = 34+13.6i
Complex number cn3 = 3.4 * cn1 = 34+13.6i
Complex number cn3 = cn1 / cn2 = 2
Complex number cn3 = cn1 = 10+4i
Complex number cn3 /= cn2 = 2
```

Testing example 2:

```
Input a complex number for cn1 : 6.3  -2.1
Complex number cn1 = 6.3-2.1i
Input a complex number for cn2 :  -2.1  1.0
Complex number cn2 = -2.1+i
Complex number cn3 = cn1 + cn2 = 4.2-1.1i
Complex number cn3 = cn1 = 6.3-2.1i
Complex number cn3 += cn2 = 4.2-1.1i
Complex number cn3 = cn1 - cn2 = 8.4-3.1i
Complex number cn3 = cn1 = 6.3-2.1i
Complex number cn3 -= cn2 = 8.4-3.1i
Complex number cn3 = cn1 * cn2 = -11.13+10.71i
Complex number cn3 = cn1 = 6.3-2.1i
Complex number cn3 *= cn2 = -11.13+10.71i
Input a real number: -2.4
Complex number cn3 = cn1 * -2.4 = -15.12+5.04i
Complex number cn3 = -2.4 * cn1 = -15.12+5.04i
Complex number cn3 = cn1 / cn2 = -2.83364-0.349353i
Complex number cn3 = cn1 = 6.3-2.1i
Complex number cn3 /= cn2 = -2.83364-0.349353i
```

Testing example 3:

```
Input a complex number for cn1 : 3.75  0.0
Complex number cn1 = 3.75
Input a complex number for cn2 : 0.0  -2.5
Complex number cn2 = -2.5i
Complex number cn3 = cn1 + cn2 = 3.75-2.5i
Complex number cn3 = cn1 = 3.75
Complex number cn3 += cn2 = 3.75-2.5i
Complex number cn3 = cn1 - cn2 = 3.75+2.5i
Complex number cn3 = cn1 = 3.75
Complex number cn3 -= cn2 = 3.75+2.5i
Complex number cn3 = cn1 * cn2 = -9.375i
Complex number cn3 = cn1 = 3.75
Complex number cn3 *= cn2 = -9.375i
Input a real number: 1.25
Complex number cn3 = cn1 * 1.25 = 4.6875
Complex number cn3 = 1.25 * cn1 = 4.6875
Complex number cn3 = cn1 / cn2 = 1.5i
Complex number cn3 = cn1 = 3.75
Complex number cn3 /= cn2 = 1.5i
```

Testing example 4:

```
Input a complex number for cn1 : 0.0 0.0
Complex number cn1 = 0
Input a complex number for cn2 : 1.0 1.0
Complex number cn2 = 1+i
Complex number cn3 = cn1 + cn2 = 1+i
Complex number cn3 = cn1 = 0
Complex number cn3 += cn2 = 1+i
Complex number cn3 = cn1 - cn2 = -1-i
Complex number cn3 = cn1 = 0
Complex number cn3 -= cn2 = -1-i
Complex number cn3 = cn1 * cn2 = 0
Complex number cn3 = cn1 = 0
Complex number cn3 *= cn2 = 0
Input a real number: 3.0
Complex number cn3 = cn1 * 3 = 0
Complex number cn3 = 3 * cn1 = 0
Complex number cn3 = cn1 / cn2 = 0
Complex number cn3 = cn1 = 0
Complex number cn3 /= cn2 = 0
```

Testing example 5:

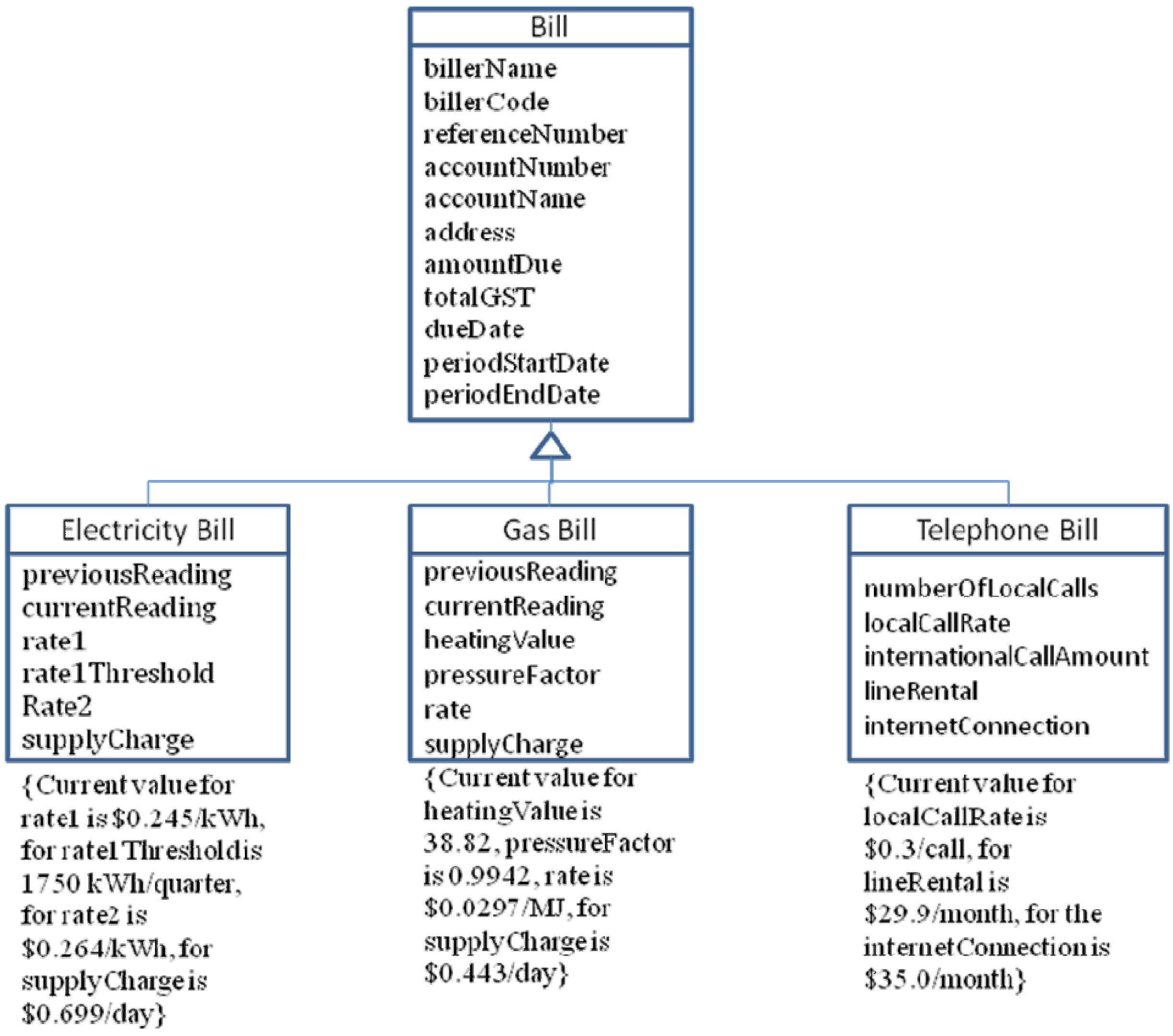
```
Input a complex number for cn1 : 2.5  -1.0
Complex number cn1 = 2.5-i
Input a complex number for cn2 : 0.0  0.0
Complex number cn2 = 0
Complex number cn3 = cn1 + cn2 = 2.5-i
Complex number cn3 = cn1 = 2.5-i
Complex number cn3 += cn2 = 2.5-i
Complex number cn3 = cn1 - cn2 = 2.5-i
Complex number cn3 = cn1 = 2.5-i
Complex number cn3 -= cn2 = 2.5-i
Complex number cn3 = cn1 * cn2 = 0
Complex number cn3 = cn1 = 2.5-i
Complex number cn3 *= cn2 = 0
Input a real number: 5.5
Complex number cn3 = cn1 * 5.5 = 13.75-5.5i
Complex number cn3 = 5.5 * cn1 = 13.75-5.5i
Exception : Division by zero.
```

Note: Your program should work on different testing data.

Task2: Class inheritance (6 marks)

In this task, we will define and implement inheritance classes.

Define the diagrams of the classes for bills below.



Define a base class **Bill** in a file **Bill.h** that contains biller's name, code, reference number, account number, account name, address, amount due, totalGST, due date, period start date and period end date. Define necessary constructors, destructor, member functions and extraction operator (>>) and insertion operator (<<) for the base class. Implement the member functions and other friend functions in a file **Bill.cpp**.

Define a derived class **ElectricityBill** in a file **ElectricityBill.h** according to the diagrams above. Implement the functions in a file **ElectricityBill.cpp**. Define and implement overloading extraction operator (>>) and insertion operator (<<) for the ElectricityBill class. **Note the rates (rate1 $0.245/kWh, rate2 $0.264/kWh), threshold (1750 kWh) and supply charge ($0.699/day) are the same for all ElectricityBill instances.**

To compute total amount due for an electricity bill, we can use the following expressions:
If total usage (*currentReading – previousReading*) is less than rate1 threshold,

*Total amount electricity = (currentReading – previousReading) * rate1;*

If the total usage is more than the threshold of rate1, we can use the following expression to compute amount of electricity

*Total amount electricity = (currentReading – previousReading) * rate1 + ((currentReading – previousReading) – rate1Threshold) * rate2;*

Then computes total supply charge,

*Total amount supply charge = (periodEndDate – periodStartDate) * supplyCharge;*

Then computes total amount due and total GST by

Total amount due = Total amount electricity + Total amount supply charge ;

*Total GST = Total amount due * 10%;*

Define a derived class **GasBill** in a file **GasBill.h** according to the diagrams above. Implement the functions in a file **GasBill.cpp**. Define and implement overloading extraction operator (>>) and insertion operator (<<) for the GasBill class. **Note the rates (heatingValue 38.82, pressureFactor 0.9942, rate $0.0297/MJ) and supply charges ($0.443/day) are the same for all GasBill instances.**

To compute total amount due for a gas bill, we can use the following expressions:

Total MJ = *(currentReading – previousReading) * heatingValue * pressureFactor;*

*Total amount gas = total MJ * rate;*

Then computes total supply charge,

*Total amount supply charge = (periodEndDate – periodStartDate) * supplyCharge;*

Then computes total amount due and total GST by

Total amount due = Total amount gas + Total amount supply charge ;

*Total GST = Total amount due * 10%;*

Define a derived class **TelephoneBill** in a file **TelephoneBill.h** according to the diagrams above. Implement the functions in a file **TelephoneBill.cpp**. Define and implement overloading extraction operator (>>) and insertion operator (<<) for the TelephoneBill class. **Note the local call rate ($0.3/call), line rental ($29.9/month) and internet connection charge ($35/month) are the same for all TelephoneBill instances.**

To compute total amount due for a telephone bill, we can use the following expressions:

Total local call = number of local calls * *local call rate;*

Then computes total amount due and total GST by

Total amount due = Total local call + international call amount + line Rental + internet connection;

*Total GST = Total amount due * 10%;*

Implement C++ main() and other functions in a file **task2Main.cpp** to display menu and get input from keyboard for bills and store bills' information in the memory (such as linkedlist or dynamic arrays) until 0 (zero) for the choice has been input. For each bill data, computes total amount, total GST due for the bill. The program will save bill's data in a text file, load bills' data from a text file and display information of loaded bills (See the Testing of the task for more details).

Testing:

Use g++ to compile the source files by

```
$ g++ –o task2 task2Main.cpp Bill.cpp ElectricityBill.cpp GasBill.cpp TelephoneBill.cpp
```

and run the program by

```
$ ./task2
```

When the program starts, it will display menu and get input data.

```
1.  Input electricity bill data;
2.  Input gas bill data;
3.  Input telephone bill data;
4.  Set electricity rates;
5.  Set gas rate;
6.  Save bill data in a text file;
7.  Load bill data from a text file;
0.  Quit.
Your choice: 1

Input electricity bill data.
Biller name: AGL
Biller code: 21345
Reference: 1234567890123445678
Account number: 12345678
Account name: Mr. Tim Smith
Address: 1 Moore Street, Wollongong, NSW 2500
Start date: 15 12 2015
```

```
End date 14 3 2016
Due date: 5 4 2016
Previous reading (kWh): 12304
Current reading (kWh): 13186

Total amount due: $279
Total GST: $27.9


1.  Input electricity bill data;
2.  Input gas bill data;
3.  Input telephone bill data;
4.  Set electricity rates;
5.  Set gas rate;
6.  Save bill data in a text file;
7.  Load bill data from a text file;
0.  Quit.
Your choice: 2

Input gas bill data.
Biller name: AGL
Biller code: 13245
Reference: 123456789012344321
Account number: 123443211
Account name: Mr. Tim Smith
Address: 1 Moore Street, Wollongong, NSW 2500
Start date: 10 12 2015
End date 16 3 2016
Due date: 30 3 2016
Previous reading (Cubic meters): 7724.5
Current reading (Cubic meters): 7796.3

Total amount due: $125.27
Total GST: $12.53


1.  Input electricity bill data;
2.  Input gas bill data;
3.  Input telephone bill data;
4.  Set electricity rates;
5.  Set gas rate;
6.  Save bill data in a text file;
7.  Load bill data from a text file;
0.  Quit.
Your choice: 3
```

```
Input telephone bill data.
Biller name: Telstra
Biller code: 33215
Reference: 123456789011112222
Account number: 11112222
Account name: Mr. Tim Smith
Address: 1 Moore Street, Wollongong, NSW 2500
Start date: 12 2 2016
End date 11 3 2016
Due date: 31 3 2016
Number of local calls: 23
International calls: 43.2

Total amount due: $115
Total GST: $11.5


1.  Input electricity bill data;
2.  Input gas bill data;
3.  Input telephone bill data;
4.  Set electricity rates;
5.  Set gas rate;
6.  Save bill data in a text file;
7.  Load bill data from a text file;
0.  Quit.
Your choice: 4

Set electricity rates.
Rate 1 ($ per kWh): 0.256
Threshold (kWh): 1800
Rate 2 ($ per kWh): 0.274
Supply charge rate ($ per day): 0.712

New rates for electricity bills have been set.


1.  Input electricity bill data;
2.  Input gas bill data;
3.  Input telephone bill data;
4.  Set electricity rates;
5.  Set gas rate;
6.  Save bill data in a text file;
7.  Load bill data from a text file;
0.  Quit.
Your choice: 5
```

```
Set gas rates.
Rate ($ per MJ): 0.0302
Heating value: 38.45
Pressure factor: 1.03
Supply charge rate ($ per day): 0.472

New rates for gas bills have been set.
```

The program will not stop until an input choice is 0 (zero).

Assume 6 bills' data have been added in. When the choice is 6, the program will save all bills into a text file.

```
1.  Input electricity bill data;
2.  Input gas bill data;
3.  Input telephone bill data;
4.  Set electricity rates;
5.  Set gas rate;
6.  Save bill data in a text file;
7.  Load bill data from a text file;
0.  Quit.
Your choice: 6

Text file name: bills.txt

6 bills have been saved.
```

The data in the text file should look like following (each bill stored in one line, start by the type: E-Electricity, G-Gas, T-Telephone):

```
E; AGL;21345;123456789012345678;12345678;Mr. Tim Smith; 1 Moore Street,
Wollongong, NSW 2500;15/12/2015;14/03/2016;05/04/2016;12304;13186;279;27.9
G;AGL;13245;123456789012344321;12344321;Mr. Tim Smith; 1 Moore Street,
Wollongong, NSW 2500;10/12/2015;16/03/2016;30/03/2016;7724.5;7796.3;125.27;12.53
T;Telstra;33215;123456789011112222;11112222;Mr. Tim Smith; 1 Moore Street,
Wollongong, NSW 2500;12/02/2016;11/03/2016;21/03/2016;23;43.2;115;11.5
…
```

```
1.  Input electricity bill data;
2.  Input gas bill data;
3.  Input telephone bill data;
4.  Set electricity rates;
5.  Set gas rate;
6.  Save bill data in a text file;
7.  Load bill data from a text file;
```

```
0.   Quit.
Your choice: 7

Text file name: bills.txt

6 bills have been loaded.
```

Then display all bills information.

```
Electricity bill:
Biller name: AGL
Biller code: 21345
Reference number: 123456789012345678
Account number: 12345678
Account name: Mr. Tim Smith
Address: 1 Moore Street, Wollongong, NSW 2500
Start date: 15/12/2012
End date: 14/03/2013
Due date: 05/04/2013
Previous reading: 12304
Current reading: 13186
Total amount due: $279
Total GST: $27.9

Gas bill:
Biller name: AGL
Biller code: 13245
Reference number: 123456789012344321
Account number: 12344321
Account name:  Mr. Tim Smith
Address: 1 Moore Street, Wollongong, NSW 2500
Start date: 10/12/2012
End date: 16/03/2013
Due date: 30/03/2013
Previous reading: 7724.5
Current reading: 7796.3
Total amount due: $125.27
Total GST: $12.53

Telephone bill:
Biller name: Telstra
Biller code: 33215
Reference number: 123456789011112222
Account number: 11112222
Account name: Mr. Tim Smith
Address: 1 Moore Street, Wollongong, NSW 2500
Start date: 12/02/2012
End date: 11/03/2013
Due date: 21/03/2013
```

```
    Number of local calls: 23
    International calls: $43.2
    Total amount due: $115
    Total GST: $11.5

    …
```

You may run the program like:

```
$ ./task2 < input4.txt
```

To check memory leak, you may run the program by

```
$ bcheck ./task2 < input4.txt
```

You can download input data files **input4.txt** and **input5.txt** for your testing.

Submission

This assignment is due by 11.59 pm (sharp) on Sunday 17 April 2016.

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 CSCI804 -a 2 task1Main.cpp ComplexNumber.h
ComplexNumber.cpp task2Main.cpp Bill.h Bill.cpp ElectricityBill.h ElectricityBill.cpp
GasBill.h GasBill.cpp TelephoneBill.h TelephoneBill.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% 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