University of Wollongong

School of Computing and Information Systems

CSCI251/851 **Advanced Programming** **Autumn 2018**

Assignment 4 **10 marks**

Submit Due: 11.59pm Week 13, Friday 1 June – (8 marks)
Demos due during your week 12 and 13 lab classes – (2 marks)
(See Submit Instructions for further info.)

Aim:

Gain experience developing programs using STL containers and inheritance.
On completion you should know how to:

- Write C++ code using STL container class objects.
- Use inheritance to facilitate programming with objects.
- Gain experience writing and debugging O-O programs incrementally.

Prerequisites:

This assignment involves writing a program to handle the billing of electricity, gas and phone accounts of a meter reading and billing company. Meter reading information on electricity, gas and phone usage is contained in the file "usage.txt". Each record begins with the service type (i.e. elect, gas or phone) followed with customer details:

BillingRecord
Supplier's name
Customer's name
Customer's address
Account balance
Days since last reading

. . . and other usage information which depend on the service type:

Electricity	**Gas**	**Phone**
Previous reading	Previous reading	Number of local calls
Current reading	Current reading	Local call rate
Rate 1	Heating value	Long distance call time
Rate 1 threshold	Rate	Long distance call rate
Rate 2	Supply charge	Line rental
Supply charge		

This information is used to calculate the customer's next bill. The processing of the billing records is complicated because some suppliers offer their customers discounts if they subscribe to more than one service from them. To solve this problem you are to use object oriented programming with inheritance to represent the different types of billing records. Polymorphism will also be used to allow all records to be processed efficiently in the one array.

You should complete this assignment according to the following steps and demo & submit your work according to the Submit and Demo instruction given at the end of this document. Each step is worth 2 marks and 2 marks is awarded for demoing your work on time in the lab. All code is to go in 5 files: **BillRecord.h**, **BillRecord.cpp, BillSystem.h, BillSystem.cpp** and **main.cpp**.

Billing System Design

The system is comprised of a BillSystem class object that own a vector of BillRecord objects in implemented in appropriately named files. The BillRecord class is a base class that has 3 derived classes for representing the 3 types of bills (elect. gas & phone). The BillSystem has public functions for reading the data file into the BillRecord vector, processing the billing information and displaying the records on the screen. Some of the functions of the BillRecord class are overridden to handle the different types of data in the derived classes.

Step 1 (2 marks)

For step 1, get the BillRecord base class working. Examine the incomplete code in the provided files and complete the function definitions of class BillRecord and BillSystem so that the customer details (only) are read into each BillRecord of the vector BRecs from “usage.txt”. Then test that main() displays the first 10 customer details on the screen, as shown below. Note: don’t forget to delete the pointers in vector BRecs in the BillSystem destructor.

```
 # Service Supplier Customer       Address                    AccBal  Days
 1 Phone   Origin   George Carter  24 Dingo St Exeter SA       27.49    28
 2 Gas     EA       Paul Scott     21 Beach Rd Barham NSW      92.98    30
 "  "         "           "                  "                   "       "
10 Gas     EA       Carol Harris   67 Broke St Milton NSW      20.45    29
```

Step 2 (2 marks)

Declare the derived BillRecord classes: ElectBillRecord, GasBillRecord and PhoneBillrecord in BillSystem.h. These classes inherit the base class BillRecord. Make sure you provide private data members for storing the relevant usage information shown on page 1. Change the BillRecord class’s private data member: AccountBalance and DaysSinceLastReading to protected to make them available to the derived classes. Also, make functions: ReadUsageInfo() and DisplayUsageInfo() into virtual functions so that they can be overridden by the base classes. See here: https://www.geeksforgeeks.org/virtual-function-cpp/

Now provide the derived classes with their versions of functions: ReadUsageInfo() and DisplayUsageInfo() to read and display their usage information appropriately.

Modify ReadFile() in BillSystem.cpp so that it reads the service type and then allocates a ElectBillRecord, GasBillRecord or PhoneBillrecord accordingly using the new operator. Then call their ReadCustDetails() and ReadUsageInfo() functions to have the data in the file read appropriately. You might have to work on the code in each ReadUsageInfo() function to get each derived class reading its file data correctly.

When you complete this step run the main() again to display 5 records showing the Customer and usage info on the screen something like this:

```
# Service Supplier Customer       Address                    AccBal Days
 1 Phone   Origin   George Carter  53 Dingo St Exeter SA       27.49   28
   (LCalls: 283, 0.14  DCalls: 245, 0.32  Rental: 0.44)
 2 Gas     EA       Paul Scott     84 Beach Rd Barham NSW      92.98   30
   (Readings: 14148, 14224  HV:38.40  Rate: 0.039 SuppC: 0.56)
 3 Elect   Alinta   Brian Crowe    61 Beach Rd Hanwood WA      18.06   31
   (Readings: 113144, 113568  R1: 0.28 R1Th: 535 R2: 0.30 SuppC: 0.86)
      "            "            "            "              "            "
```

Step 3 (2 marks)

To win more customers Dodo and Alinta provide 15% and 20% discount, respectively, to all customers who subscribe to all three services from them. To deal with this add a protected data member to the BillRecord class named Discount and a public member function named SetDiscount(float d) for setting it. Also set Discount to one in the constructor. (Note: for 15% discount set Discount to 0.85. For 20% discount set Discount to 0.8.)

There are various methods for finding which customers should have the discount applied. You can devise your own method if you wish but try to keep it efficient. The processing for this should be done in a public member function in the BillSystem class named CalDiscounts()..

One suggestion for finding the discounted records is to use a multiset with a user defined comparison object for ordering the BillRecord pointers. The comparison object should compare the BillRecords first by name and second by address. When all Dodo or Alinta records are inserted into this multiset, customers with the same name and address will be grouped and can be accessed with multiset functions: count(), lower_bound() and upper_bound(). (E.g. erase all BillRecord pointers with a count less than 3.) Once the discount pointers are found they can be dereferenced to set the discount appropriately. Test your code by uncommenting the "step-4" code in main() and print on the screen the number of discount customers Dodo and Alinta have.

Step 4 (2 marks)

Add two more public member functions to the BillSystem class named CalBills() and PrintReport(). Also add a public pure virtual function: UpdateBalance() in the BillRecord class and override this in the derived classes. UpdateBalance() should calculate the BillAmount and add this to the AccountBalance. The BillAmount is calculated as follows:

```
Elect :     C = CrntReading - PrevReading;
            P = SuppyCharge * DaysSinceLastReading;
            If (C <= R1Thld) BillAmt = (C * Rate1 + P) * Discount – AccBalance;
            Else BillAmt = (R1Thld * Rate1 + (C - R1Thld) * Rate2 + P) * Discount – AccBalance;

Gas :       C = CrntReading - PrevReading;
            P = SuppyCharge * DaysSinceLastReading;
            BillAmt = (C * HeatingValue * Rate) * Discount  – AccBalance;

Phone :     L = LocalCallRate * NumLocalCalls
            D = LDistCallRate * LDistCallTime
            P = LineRental * DaysSinceLastReading;
            BillAmt =  (L + D + P) * Discount  –  AccBalance;
```

Note: the AccountBalance should be set to zero after the BillAmount is calculated.

PrintReport() should print the first 5 records similar to Step-1 with the "AccBal" and "Days" replaced with "BillAmt". Also, print the names and addresses of all Dodo and Alinta customers who have discounts.

Submit:

Submit your code files (and the output produced (copy & paste) using the submit facility on UNIX as shown below:

```
$ submit -u login -c CSCI251 –a4 main.cpp BillRecord.h BillRecord.cpp BillSystem.h BillSystem.cpp output.txt
```

where 'login' is your UNIX login ID
Note: CSCI851 should also submit to –c CSCI251.

<u>**You must demonstrate your program in the lab during your week 12 & 13 lab classes to be awarded the demo marks (1 mark each). Work submitted and not demoed will be marked but there is no guarantee that it will be marked before the exam.**</u>

Deductions will be made for untidy work or for failing to comply with the submission instructions. Requests for alternative submission arrangements will only be considered before the due date. An extension of time for the assignment submission may be granted in certain circumstances. Any request for an extension of the submission deadline must be made to the Subject Coordinator before the submission deadline. Supporting documentation must accompany the request for any extension. Late assignment submissions without granted extension will be marked but the marks awarded will be reduced by 1 mark for each day late. Assignments will not be accepted if more than three days late.