PROG24310: Assignment N3

Evaluation: 10 points, 10% of your final grade.
Due date: December 1th, 2019 (11:30pm)

In this assignment you need to write a C++ program that simulates different pricing plans for a cellular telephone provider.

Suppose you work in the marketing department of the "CellularX" company. Your boss wants you to compare a proposed plan that offers customers a Regular plan and a Premium plan. The Premium plan is designed so that customers who make many calls save money by paying a higher monthly fee. Which plan is best for the company?

Main Requirements:

A. Create a class **Call** that contains the following information: **phone number** called and **duration** of the call (in seconds). In the real world you also want to include date/time of the call. However, this assignment is a simplified simulation.

B. Create an Abstract class **Customer** and derived classes **RegularCustomer** and **PremiumCustomer**.

The Abstract class **Customer** should contain data members common to all customers, such as customer's name, calls made (use **list** of calls – see class **Call** above), balance, etc.

In the appropriate class (**Customer, RegularCustomer**, and **PremiumCustomer**) you need to define and/or implement the following methods (with appropriate data types):

1. **setName(name)** to set the name of the customer.
2. **getName()** to get the customer's name.
3. **addCall(number, duration)** to add the phone call to the **list** of calls, where the arguments are: phone number called and duration of the call (in seconds) – see class **Call** above.
4. **setBalance(balance)** to set the balance.
5. **getBalance()** to get the balance.
6. **computeBalance()** to compute the balance based on the **list** of calls. The appropriate formula must be used.

Note: You need to figure out in what class and how you are going to define these methods. Both **RegularCustomer** and **PremiumCustomer** classes must be derived from **Customer**. Assume the initial balance is zero.

Hint: Don't forget that class **Customer** should be **abstract one**. I.e. one or more methods above must be declared in the class **Customer** using pure virtual function.

CellularX company has two types of customers: **regular customers** (class **RegularCustomer**) and **premium customers** (class **PremiumCustomer**).
The difference between them is how the balance is calculated.

Compute a Regular Customer's balance as:
`balance = monthlyfee + percall × num_calls`
where monthly fee is $5, and per call rate is $1.

Compute a Premium Customer's balance as:
`balance = monthlyfee + persec × num_secs`
where monthly fee is $40, and per minute rate is $0.02 (i.e. persec = $0.02/60).

C. In your program you need to create a simulation: randomly generate 1000 customers in each group. Randomly generate the number of calls each customers made (between 20 to 500) and randomly generate the list of calls for each customer in each group. Each group must have the same number of customers and the same number of calls.

You must create a simulation class **Simulation** that will do the simulations and print the following information (see demo run below):

- Number of customers (same for all groups).
- Number of calls (same for all groups).

For each group you need to print:

- Average duration of the call (per customer, in minutes, rounded up: 63sec = 2min).
- Average balance (per customer).
- Name of the customer with largest balance and the balance.
- Name of the customer with smallest balance and the balance.

For each randomly generated customer in each group you need to:

1. Randomly generate full customer name (4-8 chars for first name and last name). Incorrect full names such as "Xrty Ghtezuu" are acceptable.
   Note: the first letter in the first and last name must be in upper case.
2. For each randomly generated call you need to:
   1. Randomly generate the number called (assume it's US/Canada 10 digit number).
   2. Randomly generate duration of the call (from 20 seconds to 3 hours).

Your program must answer which plan is better for the company and print the average profit (see Demo Run in requirement **D**.).

D. The demo output of your program must look exactly as shown below.
Note: all the numbers below are **incorrect** and shown for demonstration purposes only!

Simulation run:

Number of customers in each group:	1000
Number of calls/per customer:	180

Regular Customers:

Average duration of the call:	80 mins
Average balance/per customer:	$77.12
Customer with largest balance:	Ygzh Dhad ($350.11)
Customer with smallest balance:	Bzzza Juil ($12.25)

Premiums Customers:

Average duration of the call:	87 mins
Average balance/per customer:	$55.88
Customer with largest balance:	Zjhp Cvned ($250.11)
Customer with smallest balance:	Aaop Bnuy ($24.25)

With 1000 regular customers the company will make $ 21316 more.

E. In the **Simulation** class you need to have **two vectors**: vector of **Regular Customers** and vector of **Premium Customers**.

- Every randomly generated customer must be added to the corresponding **vector**.
- Every customer has a **list** of randomly generated calls
- Every randomly generated call must be added to customer's **list** of calls.

Hint: Randomly generated customer is represented by an instance of the class **RegularCustomer** or **PremiumCustomer**.

In the **Simulation** class you must have at least the following methods:

1. **generateName()** to randomly generate the customer's name.
2. **generatePhoneNumber()** to randomly generate the number called.
3. **generateCallDuration()** to randomly generate the duration of the call
4. **generateCustomer(…)** to randomly generate a customer and generate his list of random calls – see requirement C above.
   Note: the same function must be used to generate "premium" and "regular" customers.

F. During simulation you need to calculate/set the balance for each customer and then at the end of the simulation in the vector of customers you need to find the ones with the largest/smallest balance.

For instance, in the **Simulation** class you can use an index of the customer with largest/smallest balance in each category of customers. Or you can sort the vector of Customers (see bonus **I**.) and instead directly access the first/last element of this vector.

G. The **main()** function in your C++ program must be as follows:

```
int main() {

     Simulation *sim = new Simulation();
     sim->printResult();
     delete sim;

    return 0;
}
```

H. In your C++ program you should:

- Declare constructors (if appropriate) and destructors (if used) in classes.
- Add any member functions / properties that you think are needed.
- Use proper Object-Oriented concepts (inheritance, polymorphism, class abstraction, dynamic dispatch, etc).
- Use proper C++ libraries. Do NOT use C libraries.
- Avoid using hardcoded values (use #define when appropriate).
- Use reasonably optimized code: minimize time & space complexity, place data on the heap, use proper methods for lists and vectors, avoid repetitive code, etc

I. (Bonus) In the **Simulation** class please sort the **vector** of customers (in each group) by their balance (from smallest to highest). This would allow you to easily find the Customer with the largest and smallest balance (see requirement G).

Submission:

- Please submit screenshot simulation.(jpg or png) of your simulation.
  It should look like Demo run in requirement D.
- Please save **main.cpp** as text file **main.txt** and upload it to the Assignment Dropbox by the Due date.
- Your submission must be unique or have references.
- Please **self-evaluate your code** in the comments section of the Assignment Dropbox.
- Please make sure your code is POSIX-compliant (i.e. works in Cygwin compiler)!
- **Late submissions are penalized 10% / day (1 point / day)!**

Grading Scheme:

- See Main requirements and also **Course_Introduction.pdf**, page 18.
- Compilation warnings are major mistakes.
- You'll get **-3 points** penalty if main requirements A, or G are not met.
- You'll get **-2 points** penalty if class Customer not abstract.
- You'll get **-2 points** penalty if you submit **.zip** file instead of .**txt** file.
- You'll get **-2 points** penalty if you don't submit **screenshot** or submit incorrect one or simulation run of your code on my computer will show completely different results.
- You'll get **zero grade** if your code doesn't compile or if you use somebody's code or idea without providing references.
- You'll receive a partial credit even if you don't implement all the requirements.