CS5403 Data Structures and Algorithms

Assignment 1

Assigned date: 09/08/2015

Due date: 09/16/2015

Please include your name in your source file

Do not use a separate header file, application file and implementation file. Instead, put all your code in a single file called hw1.cpp and submit it via the New classes course site.

Please include the return 0; statement in main() in this assignment and all future assignments.

Do not use global variables.
Do not overload operators.
Do not use friend functions.
Do not use STL.
Do not use the <vector> library.

Tasks:

Imagine you are a developing a simple prepaid phone application for a mobile network operator. Write a C++ code that implements PrepaidAccount, class which have the following:

- Attributes:
    1. phone number: 10 character String (i.e. 718-260-0000)
    2. available balance: dollar value as a floating point number

- Member Functions:
    1. Constructor: initialize an account with given phone number. The balance should be initialized to zero.
    2. AddBalance(double howMuch): adds dollars to the available balance
    3. PayForCall(int callDuration, double tariff): deducts appropriate amount of dollars from available balance. callDuration is duration in seconds and tariff is how much the call costs per minute. Don't forget to change duration from seconds to minutes in order to compute the

cost of the call properly. The function returns how much dollars are deducted from the balance. If call cost is larger than available balance, return -1 and do not make any charges.

After you implement the class, write a non-member function, compareAccounts(const PrepaidAccount a1, const PrepaidAccount a2), which returns:

- -1 : if a1's phone number is smaller than a2's phone number
- 0 : if a1's phone number is equal to a2's phone number
- 1 : if a1's phone number is larger than a2's phone number

Now write a C++ code that implements a container class for PrepaidAccount objects, called AccountStore, which should hold at most 100 account sorted with respect to their phone numbers. (Hint: Use compareAccounts function when maintaining the sorted order.) AccountStore should have the following functions:

- Constructor: Initialize AccountStore with size=0
- getSize(): returns how many accounts are currently stored
- getAccount(int i): Returns the account with i^th smallest phone number.
- removeAccount(string phoneNumber): removes the account with the given phoneNumber from container. Returns -1 if account is not found. Note that the remaining accounts should be in sorted order after removing the desired account. Don't forget to update the size.
- insertAccount(PrepaidAccount a): Inserts the given account to the appropriate position in the container. (Hint: you may need to shift several items in the array to the right to maintain sorted order) Don't forget to update the size.

Finally, write a main function to test your code. In the function generate 5 PrepaidAccount objects and 1 AccountStore object. PrepaidAccount should be initialized with random 10-digit phone numbers. For each account object:

- add random amount of dollars (between $50 and $100). Use rand() function
- Make a phone call with random duration (between 10 seconds 200 seconds) and random tariff (between .1 dollars/minute and .5 dollars/minute).
- Insert account into the AccountStore

```
(Hint: use srand(time(NULL)); and rand(); to generate random numbers)
```

Then call getAccount(int i) of the AccountStore for i=1,2,3,4,5. Print out the phoneNumber and availableBalance of the returned accounts. Note that if your AccountStore implementation is correct then the printed phone numbers should be in ascending order.

Questions:

- Analyze the best-case and worst-case running time of insertAccount() and removeAccount() methods of AccountStore class. Give your answer in Big-O notation and briefly explain. Submit your answer as another document separate from the source code.