Programming Design, Spring 2013
Homework 13

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

To submit your work, please upload the following one file to the online grading system PDOGS at http://stella.im.ntu.edu.tw/online-judgement/.

1. A CPP file for Problem 1 (to the PD015 section).

NO hard copy and NO late submission. The due time of this homework is ***1:00pm, June 16, 2013***.

Problem 1

(100 points) Consider the class `JobLinkedList` we discussed in class, whose C++ implementation of `JobLinkedList` is provided to you in the file "PDSp13_hw15_JobLinkedList.cpp". While there is an instance function `Job remove(int index)`, which removes the job currently ranked at the (`index`+1)-th position, in this homework you need to implement another function `int remove(string jobName)`, which removes those jobs whose `name` is identical to `jobName`. Please note that:

- You should return the number of jobs removed, not those jobs that are removed.
- If there is no job having its `name` identical to `jobName`, no job should be removed.
- If there are more than one job having their `name` identical to `name`, all of them should be removed.

You also need to implement a function that calculates the number of hours required to complete all existing jobs (i.e., the sum of `hour` of all jobs).

The testing data will contain lines of instructions. There are four types of instructions:

1. Inserting a new job: In this case, the line will contain a character `I`, then a white space, then the name of a job (a string with no space), then a white space, then the number of hours required for completing the job (a positive integer), then a white space, and finally an index (a nonnegative integer). When you see this instruction, all you need to do is to invoke the `bool insert(Job job, int index)` function that has been provided to you.
2. Removing an existing job by index: In this case, the line will contain a character `R`, then a white space, and finally an index (a nonnegative integer). When you see this instruction, all you need to do is to invoke the `Job remove(int index)` function that has been provided to you.
3. Removing one or some existing jobs by names: In this case, the line will contain a character `N`, then a white space, and finally a name of job (a string with no space). When you see this instruction, you should invoke the `int remove(string jobName)` function implemented by yourself. Moreover, you need to output the number of removed jobs as an integer in a single line.
4. Calculating the total number of hours required to complete all existing jobs. In this case, the line will contain a single character `C`. You need to output the the sum of `hour` of all existing jobs as an integer in a single line.

Please note that in our list, the first slot is indexed as 0, the second slot is indexed as 1, etc. An insertion operation with index i will put the new job into the $(i+1)$th slot and a removal operation with index i will remove the job occupying the $(i+1)$th slot.

Example

Suppose you are given the following testing data

```
I Tennis 1 0
I Facebook 4 1
I Calculus 3 0
I PTT 2 2
C
R 1
C
I PTT 1 3
N Calculus
I Programming 8 0
N PTT
C
```

Your output should be (without those after `//`)

```
10 // for C: 3 + 1 + 2 + 4 = 10
9  // for C: 3 + 2 + 4 = 9
1  // for N Calculus: 1 job is removed
2  // for N PTT: 2 jobs are removed
11 // for C: 8 + 3
```

Grading criteria

First of all, the TAs will open your C++ file to check whether you implemented the two functions with linked list. If this is not the case, you will get zero point. If you have done so, your program will be graded based on the following criteria:

- 100% of your grades for this program will be based on the correctness of your output. The online grading system will input a set of testing data, which includes many lines of events. Among all the events, 40 lines will require outputs. You may only see the grades of running your program on these data but cannot see the inputs and outputs. The 40 output lines count for 100 points, i.e., 2.5 points for each line.

A special note on the testing data

To allow you to get partial credits easier, the 40 lines requiring outputs will be organized as follows:

- The first twenty will all be `C` operations.
- The remaining twenty will be `C` and `N` operations.