Homework #5

RELEASE DATE: 05/12/2015

DUE DATE: 05/26/2015, **16:20** in CSIE R102/R104 and on github

As directed below, you need to submit your code to the designated place on the course website.

Any form of cheating, lying or plagiarism will not be tolerated. Students can get zero scores and/or get negative scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

Both English and Traditional Chinese are allowed for writing any part of your homework (if the compiler recognizes Traditional Chinese, of course). We do not accept any other languages. As for coding, either C or C++ or a mixture of them is allowed.

This homework set comes with 200 points and 20 bonus points. In general, every homework set of ours would come with a full credit of 200 points.

5.1 Heap and Hash

(1) (20%) Complete Exercise R-8.24 of the textbook.

(2) (20%) Complete Exercise C-8.4 of the textbook.

(3) (20%) Complete Exercise C-8.14 of the textbook.

(4) (20%) Hash function is everywhere. Use any search engine to study the term "MinHash" Explain to the TAs what it is and why it is useful. Also, cite the website that you learn the term from.

(5) (20%) Suppose there are two strings that differ by only one character. Given a magic function `postfixHash(str, k)`, which returns the hash value on the last k characters (i.e. k-postfix) of the string `str` in $O(1)$ time complexity. Describe an algorithm to find out the position that the two strings differ efficiently. Briefly discuss and justify the time complexity of your algorithm. You can assume that collision does not happen. That is, two different strings will map to two different hash codes.

(6) (Bonus 20%) Construct a perfect hash function that is efficiently computable for the following 32 standard keywords in C. You need to explain why the hash function is perfect and why it is efficiently computable to get the full bonus.

```
auto, break, case, char, const, continue, default, do, double, else, enum,
extern, float, for, goto, if, int, long, register, return, short, signed,
sizeof, static, struct, switch, typedef, union, unsigned, void, volatile,
while
```

5.2 Distributed System

With the development of technology, computers can be connected via a network and hence distributed systems become popular. A distributed system is a software system in which components located on network-connected computers communicate and coordinate their actions by passing messages. In this

way, complicated and time-consuming calculation can be divided into smaller tasks which are then assigned to the computers in the distributed system. In this homework, we would like you to construct the task queue of every computer in a distributed system using a data structure called the binomial heap.

Each computer in the distributed system has its own task queue. A task has an ID *id* and a priority value p. The larger the priority value is, the more important the task is. A task with a larger priority value should be executed first. New tasks can be assigned to any computers in the system. Also, if the system detects that the workload of the computers is not balanced, it can request one computer to give all its tasks to another. However, the system sometimes misjudges and asks a computer with fewer than w tasks to give all its tasks to another computer. Under this circumstance, the computer will refuse to give its tasks to another computer.

The reason we ask you to implement the binomial heap (`http://en.wikipedia.org/wiki/Binomial_heap`) is that it is particularly useful for the "merging" of task queues of two computers. In particular, only $O(\log n)$ time is needed to merge two size-n task queues.

To facilitate your implementation, the TAs have provided a prototype of the binomial heap. You can choose to directly implement the missing code from the prototype, or discard the prototype and implement your own **as long as all the public member functions of the BinomialHeap class can be compatibly called**.

(1) (30%) Finish/rewrite the BinomialHeap class, and describe how you test whether the data structure is correct.

(2) Implement the system for three kinds of commands below.

(a) input command: `assign cm id p`
A new task with ID *id* and priority value p is assigned to computer *cm*. You should print the following line after you put the task into the computer's task queue, where n is the number of tasks **after** the assignment.
output: `There are n tasks on computer cm.`

(b) input command: `execute cm`
Computer *cm* needs to execute the task(s) with the largest priority value in its task queue. If there are more than one tasks with such a value, please execute them all. You should print the following line after every task execution. If there are more than one tasks executed, execute the task with smaller *id* first.
output: `Computer cm executed task id.`

(c) input command: `merge cm1 cm2`
Computer *cm2* should give all its tasks to computer *cm1* if the number of tasks is larger than or equal to some parameter w. In that case,
output: `The largest priority number is now p on cm1.`
Otherwise,
output: `Merging request failed.`

Input Format: The first line contains 2 integers, c and w, separated with a white space. c is a positive integer between 1 and 10^6 that represents the number of computers in the system, where the computers are indexed $\{0, 1, \ldots, c-1\}$. w is the workload judging parameter for merging. Each computer's task queue is empty at the beginning. The next lines consists of input commands described above until the end of file, and there will be at most 10^6 such lines.

Sample Input

```
10 2
assign 2 29 100
merge 0 2
assign 2 1 300
assign 1 23 300
merge 1 2
execute 1
```

Sample Output

```
There are 1 tasks on computer 2.
Merging request failed.
There are 2 tasks on computer 2.
There are 1 tasks on computer 1.
The largest priority number is 300 on computer 1.
Computer 1 executed task 1.
Computer 1 executed task 23.
```

There are three sub-tasks that the TAs will test for this problem.

(a) (20%) If your program can perform the `assign` and `execute` operations (but not `merge` operations) correctly within the time limit (6 seconds), you get the score of this sub-task. Note that the STL priority queue (the usual heap) may be able to do so if you implement efficiently.

(b) (20%) If your program can perform 10^3 operations correctly within the time limit (6 seconds), you get the score of this sub-task. Note that the STL priority queue (the usual heap) may be able to do so if you implement efficiently.

(c) (30%) If your program can perform all the operations correctly within the time limit (6 seconds), you get the score of this sub-task. You may need the binomial heap (or other heaps) to do so.

Submission File

Please submit your written part of the homework on all problems together to R102/R104 before the deadline. For the coding part, please follow the same guide of hw2 and submit through github (while tagging your submission as `hw5`). Please **DO NOT PUT BINARY FILES** in your repository.

Your `hw5` directory of the repository should contain the following items:

- your completed/rewritten `binomial_heap.h`
- all the source code and your own `Makefile` such that `make system` would generate a program named `system` that reads the input from `stdin` and outputs to `stdout` for Problem 5.2(2).
- an optional `README`, anything you want the TAs to read before grading your code

Please make sure that your code can be compiled and run with the Makefile on CSIE R217 linux machines. Otherwise your program "fails" its most basic test and can result in ZERO!