Homework #2

TAs' email: dsata AT csie DOT ntu DOT edu DOT tw

RELEASE DATE: 03/07/2013
DUE DATE: 03/21/2013 (**Thursday**), noon

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 10 bonus points. In general, every homework set of ours would come with a full credit of 200 points.

2.1 More about C++

(1) (10%) The following function finds the maximum element and the minimum element within an array of length N.

```
void findMaxMin(int arr[], int N, int *pMax, int *pMin){
  *pMax = arr[0]; *pMin = arr[0];
  for(int i = 1; i < N; i++){
    if (arr[i] > *pMax) *pMax = arr[i];
    else if (arr[i] < *pMin) *pMin = arr[i];
  }
}
```

The function can be called with something like

```
findMaxMin(a, 100, &resMax, &resMin);
```

Rewrite the function so it works with a call like

```
findMaxMin(a, 100, resMax, resMin);
```

(*Hint: reference!*)

(2) (10%) The `sum1` below may result in a run-time error. Why? The `sum2` below does not result in a run-time error, but there may be other problems — can you think of any?

```
int& sum1(int& a, int& b){
  int c = a + b;
  return c;
}
int& sum2(int& a, int& b){
  int* pc = new int;
  *pc = a + b;
  return (*pc);
}
```

2.2 More about Arrays

(1) (10%) Do Exercise C-3.3 of the textbook. (The faster the better!)

(2) (10%) Do Exercise C-3.4 of the textbook. (The faster the better!)

(3) (10%) The size-N Pascal triangle contains all the values of C_k^n such that $0 \leq n \leq N$ and $0 \leq k \leq n$. Describe how you can store the triangle with a dense one-dimensional array with $\frac{(N+1)(N+2)}{2}$ elements. You need to describe the memory layout and the function for getting C_k^n from the array.
(*No, you do not need to consider how to compute C_k^n. We just ask you to think about the storage.*)

(4) (Bonus 10%) From the previous question, can you come up with another way of storing the triangle that take about half of the space based on the fact that $C_k^n = C_{n-k}^n$? You need to describe the memory layout and the function for getting C_k^n from the array.

2.3 Sparse Matrix Processing

You are asked to design and implement a data structure to store a very big data set of KDDCup 2011 Track 1. KDDCup is an international data mining competition, and our dear NTU team won the double-champion that year. There are 252800275 lines in the file

`/tmp2/KDDCUP2011/track1/kddcup2011track1.txt`,

which is placed on every 217 workstation from linux1 to linux14. The format of each line is:

(UserId)`TAB`(ItemId)`TAB`(Rating)`TAB`(Date)`TAB`(Time)

The data set is a log file of Yahoo! Music rating system. Each line means that an user (UserId) gives the rating score (Rating) to a music item (ItemId) at time (Time) on date (Date). You can view the data set as a super big 4D sparse matrix M with M`[UserId][ItemId][Date][Time]`=`Rating`.

Your design should support the following actions:

- `retrieve`(*u,i,d,t*): output the (Rating) that user *u* gives item *i* at date *d*, time *t*.
- `items`(*u1,u2*): output the sorted (ItemId), line by line, which corresponds to items that are rated by both user *u1* and user *u2*.
- `users`(*i1,i2,d1,t1,d2,t2*): outputs the sorted (UserId), line by line, which corresponds to users who give rating to item *i1* and item *i2* within the time interval $[(d1, t1), (d2, t2)]$.
- `club`(*r1,r2,Is*): outputs the sorted (UserId), line by line, which corresponds to users who give ratings within $[r1, r2]$ to all items in set *Is*.

The TAs will provide the desired input/output format online. You need to follow the formats so the TA can test your program with their own input files. You are allowed to use any standard libraries that you know how to use (for the questions below).
The purpose of the homework is to help you understand that designing data structures for *large data* is a non-trivial problem. We understand that you do not know many tools (yet). So please just try your best to come up with *something*. We also encourage you to be creative!

(1) (30%) Describe your design of the data structure. Emphasize on why you think the data structure would be (time-wise) efficient for the five desired actions.

(2) (20%) Compute how much time it takes to convert the raw data to your data structure. Be sure to describe the platform you are using. (The faster the better!)

(3) (20%) Calculate the exact number of bytes that your data structure consumes. (The smaller the better!)

(4) (80%) Implement the data structure you designed and write a demo program (with the input/output format) to show how your data structure performs the five desired actions. Then, briefly (≤ 20 lines) state how you test whether your implementation is correct. Furthermore, write a Makefile to compile your codes (data structure and demo program). TAs will use *make* to compile your source code and use *make run* to run your demo program on 217 workstation.

Please submit the code (data structure and demo program) as discussed below. **Homework that cannot be compiled or run correctly on the 217 workstation will not be graded and can result in ZERO**.

Submission File (Program) and Written Copy

Please upload your program as a single ZIP compressed file to CEIBA before the deadline at noon on Thursday (03/21/2013). The zip file should be like `b86506054.zip`, where the file name should be changed to your own school ID. The ZIP file should contain the following items:

- all the source code for data structure and demo program. (.h and .cpp)
- a Makefile to compile your code and run your program.
- an optional `README`, anything you want the TAs to read before grading your code

For all the problems that require illustrations, please submit a written (or printed) copy in class or to CSIE R217 before the deadline.
MEDAL USAGE: If you want to use the gold medals for this homework, please write down the number of medals that you want on the first page of your printed copy (something like "use 2 medals"). Use your medals wisely—usage cannot be retracted.