Programming Design, Spring 2015
Homework 7

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

To submit your work, please upload a PDF file for Problems 1 and 2 and two CPP files for Problems 3 and 4 (optional) to PDOGS at http://pdogs.ntu.im/judge/. Each student must submit her/his individual work. No hard copy. No late submission. The due time of this homework is ***8:00am, April 27, 2014***. Please answer in either English or Chinese.

Before you start, please read Sections 22.1–22.6 of the textbook.[1] The TA who will prepare the solution for this homework is ***Shelley Sun***.

Problem 1

(20 points; 10 points each) Consider the following program

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
  int rn = 0;
  for (int i = 0; i < 10; i++)
  {
    srand(time(0));
    rn = rand();
    cout << rn << " ";
  }
  return 0;
}
```

(a) Once you execute this program, you will (typically) see ten identical numbers. Explain why.

(b) Sometimes you still see some differences among the ten numbers. Explain why.

Problem 2

(20 points) Consider the following (imcomplete) structures

```
struct Point
{
  int x;
  int y;
};
struct Triangle
{
  Point p1;
  Point p2;
```

[1] The textbook is *C++ How to Program: Late Objects Version* by Deitel and Deitel, seventh edition.

```
  Point p3;
};
```

where `Point` is a point and `Triangle` is a triangle on the two-dimensional Cartesian plane.

(a) (5 points) Implement a global function `double area(Triangle t)` that returns the area of a given triangle.

(b) (5 points) Implement a member function `double Triangle::area()` that returns the area of a given triangle.

(c) (10 points) Use your own words to explain which implementation you prefer and why.

Write your codes on your PDF file; you do not need to submit your codes for this problem to PDOGS.[2]

Problem 3

(60 points) Mikasa is selecting courses for her next semester. Each course has a course ID, the day of that course, and its starting and ending time. Each course starts no earlier than 8 am, ends no later than 6 pm, and occupies 2, 3, or 4 continuous hours in a single day. If Mikasa wants to enroll in two courses, she must ensure that the two courses do not conflict in time. For example, if course A is at 8-11 on Monday and course B is at 10-12 on Monday, they conflict with each other; if course C is at 11-12 on Monday, then it conflicts with B but does not conflict with A. Mikasa has assigned a score to each course to represent how she prefers to (or be required to) take that course. For a required course, the score is 10; or an elective course, the score is an integer between 1 and 3. Required courses do not conflict with each other.

Given a set of courses to select, Mikasa asks the following questions:[3]

1. If she only select required courses, does she have any day with no course?
2. After selecting all required courses, is it possible to select all score-3 courses?
3. Is it possible to have at least one score-3 course on each day?

Please write a program to help Mikasa find the answers.

Input/output formats

There are 20 input files. In each file, there are $n+1$ lines of data. The first line contains a single integer $n > 0$, the number of courses. Each of the next n lines contains the information for a course. For each course, the information provided in a line are (in order) the course ID (three capital English letters), the day of that course (1 for Monday, 2 for Tuesday, ..., and 5 for Friday), the starting and ending time (8 for 8:00 am, 16 for 4:00 pm, etc.), and the score assigned by Mikasa (1, 2, 3, or 10). Two consecutive attributes are separated by a white space. For example, for a line containing

```
PDA 1 14 17 10
```

as the five attributes of a course, we know that this course `PD` is schedule at 2 pm to 5 pm on Monday. Moreover, it is a required course. You may assume that $n \leq 1000$, the starting and ending times are integers between 8 and 18, the ending time is larger than the starting time by 2, 3, or 4, and course IDs are all distinct.

[2] For some hints to calculate the area, see, e.g., http://en.wikipedia.org/wiki/Heron%27s_formula.

[3] You may be wondering whether score-1 and -2 courses play any role in answering these questions. Indeed the answer is no. They are provided to you only to make this programming assignment (slightly) harder).

Given the input file, you output three integers a, b, and c, separated with two white spaces. a is the number of days that Mikasa has no required course; b is 1 if all score-3 courses may be selected (after selecting all required courses) and 0 otherwise; c is the number of days on which no score-3 course may be scheduled (after selecting all required courses). As an example, suppose that the input file contains

```
5
PDA 1 14 17 10
ORB 2 9 12 10
STC 2 11 13 3
LKD 4 13 15 2
PME 3 9 13 3
```

in six lines, then the output file should contain

```
3 0 4
```

in one single line. Here, $a = 3$ because there are only two required courses scheduled on Monday and Tuesday, $b = 0$ because `ST` conflicts with `OR`, and $c = 4$ because Mikasa may schedule a score-3 course on only Wednesday.

What should be in your source file

You are required to use the following structure

```
struct Course
{
  char id[3];
  int day;
  int stTime;
  int endTime;
  int score;
};
```

to store the attributes of courses.[4] Each course should be stored as a `Course` variable. If you think it will help, you may add other member variables or member functions into `Course`.

Once you read n, you should declare a dynamic `Course` array by executing

```
Course* courses = new Course[n];
```

where `n` stores the number of courses. Once you do that, you will get an array of `Course` variable, each representing one course. Be aware of memory leak!

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are NOT allowed to use techniques not covered in lectures. You should write relevant comments for your codes.

Grading criteria

40 points for this program will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. For each set of input data, if your program outputs correctly without violating the space limit, you get 2 points.

5 points for this program will be based on whether you really use the structure defined above and dynamic memory allocation (in the right way).

[4] Course IDs are actually not relevant for solving this problem. However, to make your program closer to a real calendar, please still record them.

15 points for this program will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.

Problem 4 (bonus)

(20 points) Consider Mikasa's problem again. Given the same inputs as specified in Problem 3, Mikasa now wants to know, for each day, the number of hours between 8 am and 6 pm that are not covered by any course. More precisely, given an input file, you output five integers d_1, d_2, ..., and d_5, separated with four white spaces. Here, d_i is the number of hours that are not covered by any course on day i. As an example, suppose that the input file contains

```
7
PDA 1 14 17 10
ORB 2 9 12 10
STC 2 11 13 3
LKD 4 13 15 2
PME 3 9 13 3
ABF 3 13 17 3
FEG 3 12 14 1
```

in eight lines, then the output file should contain

```
7 6 2 8 10
```

in one single line. Please note that the maximum number is 10 rather than 24. Please pay attention to Tuesday. Four hours, 9-13, are covered by the two courses (including the hour 11-12 that is covered by both courses); 6 hours are not. This is why $d_2 = 6$. It does not matter whether you can enroll in ORB or not; we only care about how many hours are not covered by *any* course.

20 points for this program will be based on the correctness of your output judged by PDOGS. 10 input files, each counts for 2 points, will be given to you. You may use any technique you like.