Programming Design, Spring 2016
Homework 4

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

Please upload one PDF file for Problem 1 and two or three CPP files for Problems 2, 3, and/or 4 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 ***2:00 am, March 21, 2016***. Please answer in either English or Chinese.

Before you start, please read Sections 6.1–6.4 of the textbook.[1] Sections 6.7 and 6.8 are also helpful. If you are wondering what are "functions" in C++, you may skim through the first few pages of Chapter 5, which will formally introduced later in this semester.

The TA who generates the testing data and grades this homework is ***Parker Chiang***.

Problem 1

(20 points; 5 points each) Consider the following program:

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

int main()
{
  char c = 125;
  int a = 10;
  cout << c + a; // (a)
  c = 254;
  cout << c + a; // (b)

  int array[a]; // (c)

  return 0;
}
```

(a) What is the output of the first `cout` statement? Explain this consequence by explaining what happens to `c` when it is cast into an `int` variable.

(b) What is the output of the second `cout` statement? Explain this consequence by explaining what happens to `c` when it is cast into an `int` variable.

(c) Explain why the array `array` is created in a bad way.

(d) Modify the line

```
int array[a]; // (c)
```

into

```
int array[a] = {0}; // (d)
```

Compile the modified program and write down the error message, if any. Try to guess why.

Hint. What kind of error may occur if more than one initial value are given?

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

Problem 2

(40 points) Given a sequence of integers x, we are interested in finding whether another given sequence of integers y exists in x. For example, $(1, 2, 3)$ exists in $(3, 4, 5, 7, 1, 2, 3, 8)$ while $(5, 1)$ does not. Moreover, $(3, 5, 7)$, $(3, 2, 1)$, and $(3, 8, 3)$ also do not exist. In this example, the sequence x is call our *main sequence* while y is called our *target sequence*.

In this problem, we will write a program to complete this search task. If a target sequence does not exist in the main sentence, print out 0; otherwise, the position of the first integer in the main sequence of the first occurrence of the target sequence should be printed. The first position of the sequence is indexed as 1.

Input/output formats

There are 15 input files. In each file, there are two lines of integers. The first line contains a sequence of $n + 1$ integers n, x_1, x_2, ..., and x_n, where the last n integers form the main sequence, and the second line contains a sequence of $m + 1$ integers m, y_1, y_2, ..., and y_m, where the last m integers form the target sequence. We have $n \in \{1, ..., 1000\}$, $m \in \{1, ..., 50\}$, and x_i and y_j within -10000 and 10000, $i = 1, ..., n$, $j = 1, ..., m$. Two consecutive integers are separated by a white space.

Given these input values, your program should print out

$$\min\left\{i \in \{1, 2, ..., n - m + 1\} \middle| y_1 = x_i, y_2 = x_{i+1}, ..., y_m = x_{i+m-1}\right\}$$

if such i exists; otherwise, print out 0.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use only techniques covered so far. NO other techniques are allowed. Finally, you should write relevant comments for your codes.

Grading criteria

- 30 points 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. Each fully correct set of outputs gives you 2 points.
- 10 points 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 3

(40 points) The biggest technology-related news recently is of course about AlphaGo. In this problem, let's teach your computer how to play games! To keep it simple, let's play tic-tac-toe. To facilitate our discussion, we will label the squares on the game board according to the following figure:

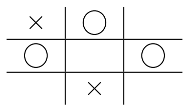
1	2	3
4	5	6
7	8	9

The top, middle, and bottom rows are labeled as rows 1, 2, and 3; the left, middle, and right columns are labeled as columns 1, 2, and 3; the diagonal made by positions 1, 5, and 9 is called diagonal 1, and the other is called diagonal 2.

Suppose that your program BetaTTT is playing tic-tac-toe with a human. BetaTTT's strategy is the following. First, you define an *objective function*

$$\max\ S_1x_1 + S_2x_2 + S_3x_3$$

for it, where x_i is the number of lines with i squares occupied by it and no square occupied by the opponent and S_i is the scores per such a line, $i = 1, 2, 3$. For example, consider the following situation:



For the cross player (he), the number of lines with one square occupied by him is 5: row 1, column 1, and diagonal 1 from the cross at position 1 and row 3 and column 2 from the cross at position 8. However, only diagonal 1 and row 3 are not occupied by any circle. Therefore, $x_1 = 2$ for him. Obviously, $x_2 = x_3 = 0$. Suppose $S_1 = 1$, $S_2 = 10$, and $S_3 = 100$, the scores for the cross player will be 2. For the circle player (she), the number of lines with one square occupied by her is 4: row 1, column 1, column 2, and column 3. As only column 3 has no square occupied by a cross, $x_1 = 1$ for her. Because row 2 is occupied by two circles but no cross, $x_2 = 1$. Finally, $x_3 = 0$. The circle player's scores are 11.

Given the objective function and given values of S_1, S_2, and S_3, a very naive strategy for BetaTTT to adopt is: Among all empty squares, choose the one that maximizes its own score after this step. If there are multiple candidate squares, choose the one with the smallest index (position 1 has the highest priority, then position 2, then position 3, ...). Consider the above example again. If BetaTTT is the cross player, it now has four choices: positions 3, 5, 7, and 9. It can be calculated that once each position is chosen, the resulting scores after this step will be 3, 12, 12, and 20. Therefore, BetaTTT will choose to put a cross at position 9.

In this problem, you will be given an unfinished tic-tac-toe game. Your BetaTTT will play as a player and implement the above strategy. Your task is to determine where to put a cross and the resulting scores.

Note. *Of course the above strategy is not so good. For the above example, obviously choosing position 5 is better than choosing position 9. One immediate way to improve this strategy is to minimize the maximum scores your opponent may get after her/his next step. By doing so, you are considering two steps from now, instead of just one step. The improvement can keep going as you extend your consideration to three steps, four steps, ..., until nine steps. The coefficients S_1, S_2, and S_3 also play roles in the performance of BetaTTT. Computer scientists need to find values of S_is to maximize the winning probability. One way is to let BetaTTT play a huge number of games, probably with computer programs implementing other strategies, with various combinations of S_is to conclude which set of values gives the best objective function. You actually have the ability to do all of these, though not required in this homework.*

There are of course more advanced methods. To further investigate this issue, you may want to study in machine learning, data mining, artificial intelligence, *or other related fields. Before that, practice to write good programs!*

Input/output formats

There are 15 input files. In each file, there is a line of integers n, S_1, S_2, S_3, y_1, y_2, ..., and y_n, where $n \in \{1, 2..., 7\}$, $S_i \in \{1, 2, ..., 1000\}$ for $i = 1, 2, 3$, and $y_i \in \{1, 2, ..., 9\}$ for $i = 1, ..., 7$. Two consecutive integers are separated by a white space. We use y_i to denote the position occupied by the circle player for $i \in \{1, 3, 5, 7\}$ or that occupied by the cross player for $i \in \{2, 4, 6\}$. For example, an input line

```
5 1 10 100 2 1 4 8 6
```

means that the circle player occupies squares 2, 4, and 6, and the cross player occupies squares 1 and 8. This is exactly the situation illustrated by the above example.

Given this input, your program should continue to decide the square to occupy in the next step. If n is odd, BetaTTT is the cross player; otherwise, it is the circle player. Your program should follow the strategy defined in this problem and identify the square that maximizes its own scores after this step. The objective function should adopt S_is given in the input as the coefficients. Print out the position to occupy and the resulting scores, separated by a white space. For the above example, the output should be

```
9 20
```

Do not forget that if multiple squares all result in the same maximum scores, occupy the one with the smallest index.

Note. *Though not so meaningful, it is possible that a testing instance has already been finished, i.e., had a winner. For example, it is possible for you to get*

```
5 1 10 100 1 2 4 5 7
```

as the input. The expected output is indeed

```
8 100
```

though the opponent has won the game already.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use only techniques covered so far. NO other techniques are allowed. Finally, you should write relevant comments for your codes.

Grading criteria

- 30 points 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. Each fully correct set of outputs gives you 2 points.
- 10 points 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 bonus points) In this problem, we improve your BetaTTT in Problem 3 to GammaTTT by considering one more step. To choose your next step, minimize the maximum scores that your opponent may obtain in her/his next step. In the example presented in Problem 3, the cross player following the new strategy should choose position 5, otherwise the circle player will get at least 100 points. By choosing position 5, the cross player can limit the circle player's maximum scores to be 10 (by choosing either position 3 or position 9).

There are 10 input files for this problem. The input format is exactly the same as that for Problem 3. Your program should output GammaTTT's next step and the minimized maximum scores that the opponent may obtain in her/his next step. The two integers should be separated by a white space. For the above example, the output should be

```
5 10
```

For this problem, you are allowed to use any technique. All 20 points 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. Each fully correct set of outputs gives you 2 points.