Programming Design, Spring 2013
Lab Exam 3

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

In this exam, there are three problems. You need to write a C++ program for each problem. 100% of your grades for this program will be based on the correctness of your output. The online grading system will input in total 50 sets of testing data and then check your outputs. These 50 sets count for 100 points, i.e., 2 points for each set. Before the due time of the exam, you may upload your programs multiple times. Only the last one you upload will be graded. Nevertheless, you will not see your scores during the exam.

Please note that your program will be submitted to the online grading system. Therefore, your programs must be able to accept multiple lines of input and stop when the input ends. In short, your programs should be structured in the same way as in the homework.

Problem 1

(30 points) In a club (or class, company, society, etc.) with many members, it is usual that A knows B but B does not know A. Some people are more "popular", i.e., known by most people but only know few people. In this problem, you need to find those most popular people.

The input contains several lines of values. In each line, $2m + 2$ positive integers n, m, u_1, v_1, u_2, v_2, ..., and u_m, v_m will be given, where n is the number of people in the club and m is the number of "social links". Between two values there is a white space. Each pair of u_i and v_i represents a relationship between person u_i and v_i: Person u_i knows person v_i. For each person, we define the celebrity factor as the number of people knowing her minus the number of people she knows. You need to find out those people who have the highest celebrity factor. As an example, the input

```
4 3 3 1 4 3 2 1
```

tells us that there are three links among four people. The first two numbers 3 and 1 means "person 3 knows person 1", the third and fourth numbers 4 and 3 means "person 4 knows person 3", and the last two numbers 2 and 1 means "person 2 knows person 1". In this example, the celebrity factors of persons 1, 2, 3, and 4 are 2, -1, 0, and -1, respectively. Therefore, the output should be a single number `1`. If multiple people all have the highest celebrity factor, all of them should be printed out in the ascending order with white spaces separating them.

Below are some examples:

- Input: `5 5 4 2 1 5 5 2 2 5 3 2`. Output: `2`.
- Input: `2 1 2 1`. Output: `1`.
- Input: `5 6 1 2 1 3 4 2 5 3 2 5 3 4`. Output: `2 3`.

Problem 2

(20 points) Continue from Problem 1, now we still want to find the most popular people. However, now each set of input data will only contain $2m$ positive integers: u_1, v_1, u_2, v_2, ..., and u_m, v_m. All others remain the same. As an example, the input

```
3 1 4 3 2 1
```

describes exactly the same environment as that described in Problem 1. Therefore, the output should again be a single number `1`.

In short, only the input format is changed. Please note that (1) the number of people involved in each set of testing data is the maximum value among all given values, (2) such a number cannot be found without looking at all the values, and (3) as the size of each set of testing data may vary, you need to first read the whole line as a C++ string and then split the string according to white spaces.

Below are some examples:

- Input: `4 2 1 5 5 2 2 5 3 2`. Output: `2`.
- Input: `2 1`. Output: `1`.
- Input: `1 2 1 3 4 2 5 3 2 5 3 4`. Output: `2 3`.

Problem 3

(50 points) In this problem, you need to implement the bingo game whose table size is not limited to five by five. For each set of testing data, the line contains $m+3$ values n, s, m, x_1, x_2, ... and x_m. The first value n means the table size is n by n. The second value s is the random number seed that will be used to construct the table. The third value m tells you how many cells should be boxed. The last m values indicate those cells that should be boxed. Between two consecutive values there is a white space. Your task is to output the number of complete rows, number of complete columns, and number of complete diagonals, separated with white spaces.[1]

The table should be set up according to n and s in the following steps. While the description is for general n and s, we also use $n = 3$ and $s = 20$ to illustrate an example:

1. An array of length n^2 is created so that the ith element is initialized to i, $i = 1, 2, ..., n^2$. If $n = 3$, the array is $(1, 2, 3, 4, 5, 6, 7, 8, 9)$.
2. The seed s is used as the argument of `srand()` to generate n^2 random numbers r'_i, $i = 1, ..., n^2$. If $n = 3$ and $s = 20$, the nine random numbers are 103, 26079, 18073, 24951, 18538, 24795, 5078, 6508, and 13002.
3. The n^2 random numbers are used to generate n^2 "derived random numbers" $r_i = r'_i \mod n^2 + 1$. If $n = 3$ and $s = 20$, the nine derived random numbers are 5, 7, 2, 4, 8, 1, 3, 2, and 7.
4. For i from 1 to n^2, elements i and r_i are exchanged. Suppose $n = 3$ and $s = 20$. In this case, first elements 1 and 5 are exchanged and the array becomes $(5, 2, 3, 4, 1, 6, 7, 8, 9)$, then elements 2 and 7 are exchanged and the array becomes $(5, 7, 3, 4, 1, 6, 2, 8, 9)$, then elements 3 and 2 are exchanged and the array becomes $(5, 3, 7, 4, 1, 6, 2, 8, 9)$, ..., and finally elements 9 and 7 are exchanged and the element becomes $(6, 1, 2, 4, 8, 5, 9, 3, 7)$.
5. An n by n table is created so that the intersection of row j and column k is the $n(j-1)+k$ element of the array, $j = 1, ..., n$, $k = 1, ..., n$. If $n = 3$ and $s = 20$, the table will be

$$\begin{matrix} 6 & 1 & 2 \\ 4 & 8 & 5 \\ 9 & 3 & 7 \end{matrix}$$

Suppose in a set of testing data the input is `3 20 5 4 6 7 8 9`, values 4, 6, 7, 8, and 9 should be boxed and the output should be `0 1 1`.

Among the 25 sets of testing data, the first fifteen sets will be seven by seven (i.e., $n = 7$). The last ten sets will have different table sizes.

[1] As in the homework, a row/column/diagonal is complete if all the cells on that row/column/diagonal are boxed.