

ACM International Collegiate
Programming Contest 1999/Y2K
Sponsored by IBM

Arab and African Regional Contest

Supported by Omnidata, RAM, BP and EMO

Al Akhawayn University in Ifrane, Morocco

November 21st, 1999

**This problem set should contain 7 problems and 9 numbered pages
(This one included)**

ACM International Collegiate Programming Contest 1999/Y2K Arab and African Regional Contest

Problem 1

Converter

Source: conv. (c|cpp | java)

Input: conv.in

Output: conv.out

Imagine that mankind was to fight against some race that can not understand numbers. To provide secure communication, all messages exchanged by humans must be encoded as numbers. A simple way of doing this is to represent each word as a code number.

Our code is very simple. A word consists of one to twenty lowercase letters. Starting with 1, words are to be encoded in alphabetical order. Part of the list is shown below.

a	1
...	
z	26
aa	27
ab	28
...	
snowfall	157,118,051,752
...	
////////////////////	20,725,274,851,017,785,518,433,805,270

Your program is to create a table that encodes each word and decodes each number found in the input file.

Input

The input file consists of a list of words and numbers, one per line, followed immediately by a line containing only a single asterisk (*). There are no leading or trailing spaces. A number consists only of the decimal digits 0 through 9. A word consists of one to twenty lowercase letters.

Output

For each word or number in the input file, the output file is to contain a single line. On each line, the word begins in column 1 and its code number begins in column 25. Numbers that have more than three digits must be separated by commas at thousands, millions, and so forth.

Sample Input

```
29697684282993
transcendental
computationally
28011622636823854456520
////////////////////
*
```

Sample Output

elementary	29,697,684,282,993
transcendental	51,346,529,199,396,181,750
computationally	232,049,592,627,851,629,097
prestidigitation	28,011,622,636,823,854,456,520
////////////////////	20,725,274,851,017,785,518,433,805,270

ACM International Collegiate Programming Contest 1999/Y2K
Arab and African Regional Contest

Problem 2

Factorial

Source: fact. (c| cpp | java)

Input: fact.in

Output: fact.out

A palm-reader knows that the happiness of a person depends on the frequency of certain digits in the factorial of the day-in-year number of the person's date of birth. The day-in-year number will be from 1 (January 1) to 366 (December 31st in a leap year). From a course in advanced numerology you know that 366! has 781 digits.

As a personal favor to your roommate, you have agreed to provide his mother with a program that reads day-in-year number and then displays the frequencies of all digits from 0 to 9.

Input

The input data is a list of day-in-year numbers followed by zero, which signals the end of the input file.

Output

The output for each day-in-year number should appear as follows, except that spaces should replace the squares. Frequencies must be right justified.

```
100! --
(0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
(0) 0 (1) 0 (2) 0 (3) 0 (4) 0
(5) 0 (6) 1 (7) 0 (8) 0 (9) 0
(0) 2 (1) 0 (2) 1 (3) 1 (4) 1
(5) 0 (6) 0 (7) 0 (8) 0 (9) 0
(0) 30 (1) 15 (2) 19 (3) 10 (4) 10
(5) 14 (6) 19 (7) 7 (8) 14 (9) 20
```

Sample Input

```
3
8
100
0
```

Sample Output

```
3! --
(0) 0 (1) 0 (2) 0 (3) 0 (4) 0
(5) 0 (6) 1 (7) 0 (8) 0 (9) 0
8! --
(0) 2 (1) 0 (2) 1 (3) 1 (4) 1
(5) 0 (6) 0 (7) 0 (8) 0 (9) 0
100! --
(0) 30 (1) 15 (2) 19 (3) 10 (4) 10
(5) 14 (6) 19 (7) 7 (8) 14 (9) 20
```

ACM International Collegiate Programming Contest 1999/Y2K
Arab and African Regional Contest

Problem 3
Job Processing

Source: job. (c|cpp |java)
Input: job.in
Output: job.out

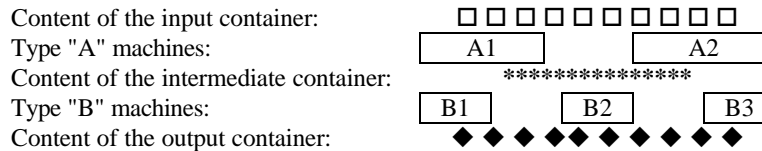


Figure 1. 2 type A machines performing operation A, and 3 type B machines performing operation B.

A factory is running a production line. Two operations have to be performed on each job: first operation "A", then operation "B". There is a certain number of machines capable of performing each operation. Figure 1 shows the organization of the production line that works as follows. A type "A" machine takes a job from the input container, performs operation "A" and puts the job into the intermediate container. A type "B" machine takes a job from the intermediate container, performs operation "B" and puts the job into the output container. All machines can work in parallel and independently, and the size of each container is unlimited. The machines have different performance characteristics and different processing times. All jobs are identical, and N jobs are available at time 0.

You are required to compute **1.** the earliest time operation "A" can be completed for all N jobs (call this subtask 1). **2.** the minimal amount of time needed to perform both operations (A and B) on the N jobs (call this subtask 2).

Input

The input file contains positive integers in five lines. The first line contains the number of jobs ($1 \leq N \leq 1000$). On the second line, the number M1 of type "A" machines ($1 \leq M1 \leq 30$) is given. In the third line there are M1 integers, the job processing times of each type "A" machine. The fourth and the fifth line contain the number M2 of type "B" machines ($1 \leq M2 \leq 30$) and the job processing times of each type "B" machine, respectively. The job processing time is measured in units of time. Each processing time is at least 1 and at most 20.

Output

Your program should write two lines to the output file. The first line should contain one positive integer: the solution of subtask 1. The second line should contain the solution of subtask 2.

Sample Input

```
5
2
1 1
3
3 1 4
```

Sample Output

```
3
5
```

ACM International Collegiate Programming Contest 1999/Y2K
Arab and African Regional Contest

Problem 4

Rectangular Patterns

Source: patt. (c | cpp | java)

Input: patt.in

Output: patt.out

Imagine an m by n (m rows by n columns) rectangle divided into squares as shown in Fig 1a below. Each location will be numbered sequentially so that the upper left square is 1, the square to its right is 2 and the bottom right is m times n .

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

Fig 1a

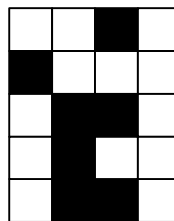


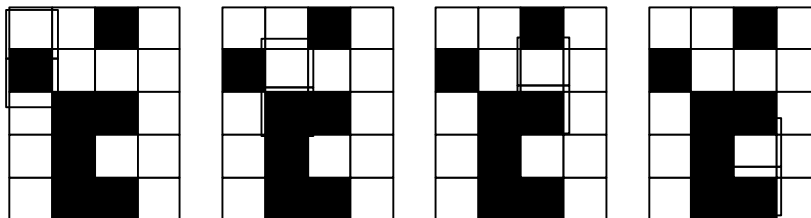
Fig 1b

Fig 1b shows a rectangle with some of the squares filled in.

Now consider the p by q (p rows, q columns) sub-region starting at the upper left corner of fig 1b. For example if $p = 2$ and $q = 1$ then we get following sub-region:



This sub-region occurs in four different locations of the original rectangle.



You must write a program that will read in a rectangle and a sub-region and print out how many times the sub-region occurs in the original rectangle. You should not consider rotations of the sub-region. Note that the initial sub-region is always selected from the upper left hand corner.

Input

The first line contains four numbers giving the values of the dimensions of the rectangle and the sub-region, m , n and p , q respectively where, $(1 \leq m \leq 20)$, $(1 \leq n \leq 20)$, $(1 \leq p \leq m)$ and $(1 \leq q \leq n)$ respectively. The next line contains the number of black squares, followed by a list of their locations as described in figure 1.a, one per line.

Output

The output will be the number of times the sub-region occurs in the original rectangle including the original top left region.

Example 1

Sample Input

5 4 2 1
7
3
5
10
11
14
18
19

Sample Output

4

Note that this example corresponds to fig 1b.

Example 2

Sample Input

4 5 2 2
2
2
5

Sample Output

2

ACM International Collegiate Programming Contest 1999/Y2K
Arab and African Regional Contest

Problem 5

Windows

Source: win. (c | cpp | java)

Input: win.in

Output: win.out

A “windowing system” is a software package that manages the representation of multiple “windows” simultaneously on the same computer screen. The windowing system must process all “events” that occur -- the pressing of a mouse button, the movement of the mouse, the pressing of a keyboard key, and so on. Some of the difficult events that are to be handled occur when one window overlaps another window, so that only part of one of the windows is drawn. (Which window is incompletely drawn, of course, depends on which window is “behind” the other one). This is usually represented by assigning a “depth” to each window, with a greater depth indicating that a window is “behind” a window of smaller depth. You may assume that no two windows have the same depth. You are to implement this part of a windowing system.

Input

The input consists of representations for a set of up to 50 windows. Each window is represented by six integer values on a single line separated by blank spaces. The six values represent the window id, the x and y screen coordinates of the lower left corner of the window, the width of the window, the height of the window, and the depth of the window.

Output

The output should be a series of statements about hidden window areas. Each line will describe a single rectangular area of one window hiding another window, in the form:

Window ___ hides window ___ from ___,___ to ___,___.

All of the hidden areas represented by the input should be listed.

Sample Input

```
11 10 10 100 50 1
21 50 0 30 20 5
35 80 40 50 50 10
```

Sample Output

```
Window 11 hides window 21 from 50,10 to 80,20.
Window 11 hides window 35 from 80,40 to 110,60.
```

ACM International Collegiate Programming Contest 1999/Y2K
Arab and African Regional Contest

Problem 6

Sequence

Source: seq. (c | cpp | java)

Input: seq.in

Output: seq.out

Let n , p , and q be positive integers with $n > p+q$. Let x_0, x_1, \dots, x_n be a sequence of integers satisfying the following conditions:

(a) $x_0 = x_n = 0$;

(b) $\forall i, 1 \leq i \leq n$, either $x_i - x_{i-1} = p$ or $x_i - x_{i-1} = -q$.

For each such sequence, it can be shown that there exists a match pair (i,j) of indices with $i < j$ and $(i, j) \neq (0,n)$ such that $x_i = x_j$. The first match pair is the one that has the least i .

You are to write a program to calculate these sequences together with their first match pairs as shown below.

Input

A file containing the three integer values of p , q , and n , all on the same line.

1 2 6

Output

All valid sequences of integers, one per line with their first match pair, as formatted below.

0 -2 -4 -3 -2 -1 0 match at (1,4)
0 -2 -1 -3 -2 -1 0 match at (1,4)
0 1 -1 -3 -2 -1 0 match at (2,5)
0 -2 -1 0 -2 -1 0 match at (0,3)
0 1 -1 0 -2 -1 0 match at (0,3)
0 1 2 0 -2 -1 0 match at (0,3)
0 -2 -1 0 1 -1 0 match at (0,3)
0 1 -1 0 1 -1 0 match at (0,3)
0 1 2 0 1 -1 0 match at (0,3)
0 1 2 3 1 -1 0 match at (1,4)
0 -2 -1 0 1 2 0 match at (0,3)
0 1 -1 0 1 2 0 match at (0,3)
0 1 2 0 1 2 0 match at (0,3)
0 1 2 3 1 2 0 match at (1,4)
0 1 2 3 4 2 0 match at (2,5)

ACM International Collegiate Programming Contest 1999/Y2K
Arab and African Regional Contest

Problem 7

Activity Scheduling

Source: sched. (c| cpp | java)

Input: sched.in

Output: sched.out

Activity scheduling can be a difficult task. Suppose you have a single lecture hall in which you would like to schedule as many activities as possible. Each activity has a starting time s_i and a finishing time f_i specified as nonnegative integers where $s_i < f_i$. Such an activity would need the hall from time s_i up to but not including time f_i .

For example, an activity that we would like to schedule from time 70 up to an including time 149, would have $s_i = 70$ and $f_i = 150$.

Input

An activity is represented by a line with two integers, the start time followed by the end time. A scheduling problem is represented by a line that contains the number of activities followed by the list of activities. The input file consists of a line that contains the number of scheduling problems followed by the list of scheduling problems.

Output

Your program should find the maximum number of activities that can be scheduled for each scheduling problem in the input file. Your output should consist of a line for each scheduling problem that contains a single integer representing the maximum number of activities that can be scheduled.

Sample Input

```
2
5
10 13
2 4
3 12
4 6
0 5
3
0 5
1 3
3 10
```

Sample Output

```
3
2
```