



Association for Computing Machinery
Hong Kong Chapter
Collegiate Programming Contest 2003

Venue: The University of Hong Kong

Date: June 24, 2003.

Time: 2:00pm to 6:00pm

Number of questions: 6

Number of non-cover pages: 14

Association for Computing Machinery
Hong Kong Chapter

Question 1: Computing Determinant

Linear algebra is one of the main topics in engineering mathematics. It includes the theory and application of linear systems which help solving many difficult problems. Matrix is a tool commonly used in linear algebra, and very often we need to calculate the *determinant* of a matrix. The determinant of a $n \times n$ matrix \mathbf{A} is denoted by

$$D = \det \mathbf{A} = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \cdot & \cdot & \dots & \cdot \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$$

and is defined for $n = 1$ by

$$D = a_{11}$$

and for $n \geq 2$ by

$$D = a_{j1}C_{j1} + a_{j2}C_{j2} + \dots + a_{jn}C_{jn}$$

for $j = 1, 2, \dots$, or n , or

$$D = a_{1k}C_{1k} + a_{2k}C_{2k} + \dots + a_{nk}C_{nk}$$

for $k = 1, 2, \dots$, or n , where

$$C_{jk} = (-1)^{j+k} M_{jk}$$

and M_{jk} is the determinant of the submatrix of \mathbf{A} obtained by deleting the j th row and k th column of \mathbf{A} .

You are required to write a program to compute the determinants of some input matrices. Your program should read in a file containing a few groups of lines, with each group containing information about one square matrix. Different groups are separated by a blank line in the input file. The first line of each group contains an integer n that specifies the dimension of the matrix. Following that are n lines that contain the entries for the n rows of the matrix. Each line contains n values separated by a space. Your program should output a file that contains the determinants of the input matrices. Each line should contain the determinant of one matrix according to the input order.

The input file is named "in1.txt". The result is to be printed to an output file "out1.txt". Your program should be named "q1.c" for C and "q1.java" for Java. The compiled program should be named "q1.exe" and "q1.class" respectively. In your program, no error checking is required for input.

Sample input:

2
4 3
2 5

2
10 15
1 0

Sample output:

14
-15

Question 2: Filling Hexagons

Figure 1 shows a lattice formed by 19 hexagons. There are totally $19!$ ways to fill the numbers 1 to 19 into these 19 hexagons. By adding all numbers which are in a line, we will have 15 sums as illustrated in the figure.

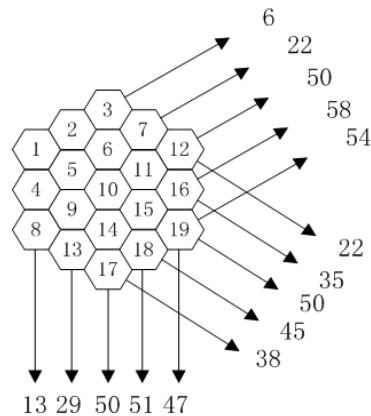


Figure 1: A lattice formed by 19 hexagons

It is easy to see that there is only one way to fill the hexagons that gives the same sum series as in the figure. Since the maximal 3 numbers should be filled in the line which have the sum 54 and so on, the hexagons can be uniquely filled. Nonetheless, given the sum series, it is not always possible to recover the filling pattern of the hexagons because there are multiple ways to fill the hexagons that give the same sum series. For example, the figure below illustrated two such filling patterns.

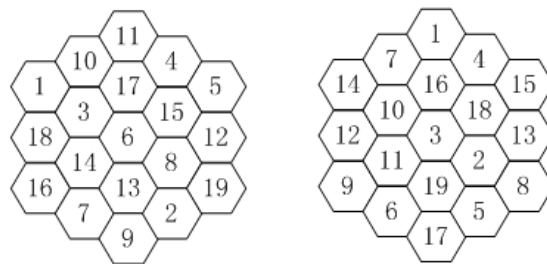


Figure 2: Two filling patterns that give the same sum series

You are required to write a program to determine the number of possible ways to fill the hexagons that give the same sum series as the input hexagons. Your program

should read in an input file consists of lines each containing 19 integers that specify one filling pattern. The first number of each line is to be filled in the position that contains the number 1 in Figure 1, the second number is to be filled in the position that contains the number 2, and so on. Your program should output a file that contains the number of possible ways to fill the hexagons for each input pattern. Each line should contain the answer to one pattern according to the input order..

The input file is named "in2.txt". The result is to be printed to an output file "out2.txt". Your program should be named "q2.c" for C and "q2.java" for Java. The compiled program should be named "q2.exe" and "q2.class" respectively. In your program, no error checking is required for input.

Sample input:

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

Sample output:

```
1
1
```

Question 3: Drawing Dendrograms

This question is about plotting trees in text mode. This task shares some common difficulties with plotting trees in graphics mode, therefore it is a foundation for more advanced plotting.

Motivation: a popular data-mining technique is called hierarchical clustering, in which similar objects are merged to form clusters in an iterative fashion. The final result can be represented as a *dendrogram*, which is an ordered binary tree (i.e., each internal node has two children, and the left-right order of the children matters). For simplicity, here we do not consider other features of dendrograms.

You are required to write a program to plot the dendrogram to the an ascii file. You can ignore the details about hierarchical clustering, and focus on the plotting of the dendrogram. Your program should read in a file containing groups of lines, with each group containing information about one dendrogram. Different groups are separated by a blank line in the input file. Each line contains the names of 3 nodes separated by two tabs, which specifies the children of an internal node in the following format:

```
<parent node name><tab><first child name><tab><second child name>
```

The name of a node is a unique case-sensitive string. The name has no meaning to the structure of the tree. For instance, whether a node is the root node cannot be determined from the node name. Also, there is no rule governing the order of the lines.

Your program should plot the dendrogram in the following format:

- It has a 90-degree left rotated orientation such that the root is on the left.
- Each line contain one node.
- The first child is plotted first (i.e., it is the right child).
- Each node is preceded by the symbol “--|” (two dashes followed by a vertical stroke).
- Siblings are connected by the symbol “|” (a vertical stroke).
- No extra space is plotted.

Your program should output the dendrograms according to the input order. For the sake of answer checking, there should be no blank line before the first dendrogram and exactly one blank line after each dendrogram.

The input file is named "in3.txt". The result is to be printed to an output file "out3.txt". Your program should be named "q3.c" for C and "q3.java" for Java. The compiled program should be named "q3.exe" and "q3.class" respectively. In your program, no error checking is required for input.

Sample input:

```
I1      L0      L1
I2      L2      L3
I3      I1      I2
I4      I3      L4
```

Sample output:

```
      --|L0
      --|I1
      | --|L1
      --|I3
      | | --|L2
      | --|I2
      |  --|L3
      --|I4
      --|L4
```

Question 4: Trading Game

In a trading game, three marbles will be distributed to all players at the beginning. Each marble is marked with a number which ranges from 1 to N , where N is the number of players involved. There are three marbles of each marker value in the whole collection of $3N$ marbles. The goal of a player is to collect three marbles with the same marker by exchanging a marble with another player. There are no restrictions on the choice of the marker value of the marble during the exchange. However, the trade is valid only if, for each trader, the sum of his/her marbles' marker values after the trade is greater than the sum before, or the number of marbles with the same marker is increased through this exchange (see the table below). When all players finish their collections, the game is ended.

Case	Player	Before exchange	After exchange	Trade
A	1	2, 4, 3 (sum = 9)	2, 5, 3 (sum = 10)	Valid
	2	1, 4, 5 (sum = 10)	1, 4, 4 (sum = 9)	
B	1	2, 4, 3 (sum = 9)	2, 1, 3 (sum = 6)	Invalid
	2	1, 4, 5 (sum = 10)	4, 4, 5 (sum = 13)	

You are required to write a program to calculate the minimum number of trades involved to finish the game. Your program should read in a file containing a few groups of lines, with each group containing information about a single game. Different groups are separated by a blank line in the input file. For each group, the first line contains an integer specifying the number of players involved. Following that are lines that specify the initial marbles marker values of each player. Your program should output a file that contains the minimum number of trades for each game.

The input file is named "in4.txt". The result is to be printed to an output file "out4.txt". Your program should be named "q4.c" for C and "q4.java" for Java. The compiled program should be named "q4.exe" and "q4.class" respectively. In your program, no error checking is required for input and you can assume that the maximum number of players will not exceed 1000.

Sample input:

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

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

Sample output:

```
4
0
```

Question 5: Rolling Slab

Consider an $n \times n$ maze with obstacles specified by an “X”. An example of a 9×9 maze with three obstacles is shown in Figure 3(a). You are now given a slab, a block of dimension $1 \times 2 \times 3$, which is initially placed on the maze at the position labelled “s”. You are now asked to find a sequence of rolling operations of the slab on the maze so as to move the slab to the destination labelled “t”. An example is shown in Figure 3(b). One possible sequence of rotation is shown in Figure 4, i.e., rolling east, then north and then east.

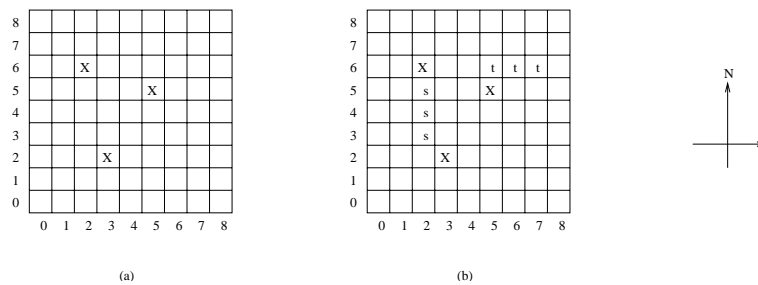


Figure 3: An example of a maze with obstacles

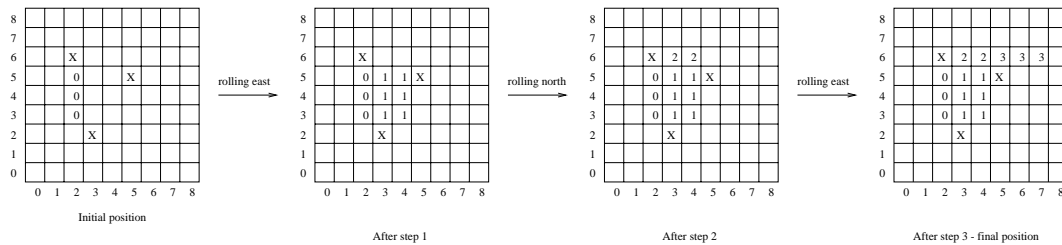


Figure 4: An example of a sequence of rotation from the source to the destination

In the input file, you are given the dimension n of the maze, the number of obstacles m , the coordinates (note that the coordinates start from $(0,0)$) of the obstacles $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$, the original position of the slab on the maze (a_s, b_s) and (c_s, d_s) where (a_s, b_s) is the upper left corner and (c_s, d_s) is the lower right corner of the rectangular space occupied by the slab initially, and the destination position (a_t, b_t) and (c_t, d_t) where (a_t, b_t) is the upper left corner and (c_t, d_t) is the lower right corner of the rectangular space occupied by the slab finally. The input will be given in the following format:

n
 m
 (x_1, y_1)
 (x_2, y_2)
 \dots
 (x_m, y_m)
 $(a_s, b_s)(c_s, d_s)$
 $(a_t, b_t)(c_t, d_t)$

In the output, you should print out a sequence of rolling operations of the slab in order to move it from the starting position to the destination. For the example in Figure 4, the input will be:

9
3
(2, 6)
(3, 2)
(5, 5)
(2, 5) (2, 3)
(5, 6) (7, 6)

while one feasible solution is:

3
E
N
E

where the first number specifies the number of steps while the sequence below specifies the direction of the rolling operations. “E” represents rolling east, “S” represents rolling south, “W” represents rolling west and “N” represents rolling north.

Your program should read in a file containing a few groups of lines, with each group containing information about one rolling slab problem. Different groups are separated by a blank line in the input file. Your program should output a group of lines, each group, separated by a blank line, corresponds to the solution of one rolling slab problem in the same order as in the input file. Your program should use *less than 1 minute* to terminate.

The input file is named “in5.txt”. The result is to be printed to an output file “out5.txt”. Your program should be named “q5.c” for C and “q5.java” for Java. The compiled program should be named “q5.exe” and “q5.class” respectively. In your program, no error checking is required for input.

Sample input:

```
9
3
( 2, 6 )
( 3, 2 )
( 5, 5 )
( 2, 5 )
( 2, 3 )
( 5, 6 )
( 7, 6 )
```

Sample output:

```
3
E
N
E
```


respectively. The mean of the region populations, μ , is calculated as

$$\mu = \frac{\sum_{\rho \in \{A,B,C,D\}} pop(\rho)}{4}$$

and the variant is

$$\frac{1}{3} \sum_{\rho \in \{A,B,C,D\}} (pop(\rho) - \mu)^2$$

For the example in Figure 5, the best location is at (1,0), where the mean of region population is 25 and the variant is 416/3, that is at a minimum.

You are required to write a program to compute the best location for the supermarket that produces the minimum population variant among the four regions. Your program should read in a file containing a few groups of lines, with each group containing information about one city. Different groups are separated by a blank line in the input file. The first line of each group contains an integer, which is the size s of the city. From the second line onwards, each line carries “ s ” integers representing populations for each row of districts. Totally there are $s + 1$ lines in each group. All numbers in a line are separated by a space. You can assume the maximum city size is 1024 and the population of each district is not more than 255. Up to this maximum limit, your program is expected to deliver an exact result *within 3 seconds*.

The input file is named “in6.txt”. The result is to be printed to an output file “out6.txt”. Your program should be named “q6.c” for C and “q6.java” for Java. The compiled program should be named “q6.exe” and “q6.class” respectively. In your program, no error checking is required for input.

Sample input:

```
4
2 5 5 6
7 3 3 7
5 7 5 5
20 15 3 2
```

```
5
0 0 0 132 225
0 0 83 82 144
0 90 241 187 233
100 179 166 219 255
135 12 62 255 255
```

Sample output:

```
1 0
2 3
```