MŰEGYETEM 1782

5th BME International

24-hour Programming Contest

Qualifying Round
Problem Set

March 5, 2005

http://www.challenge24.org

The main sponsor of the event is Fornax Co.

International Platinum Sponsor

Silver Grade Sponsor

John von Neumann Computer Society, IEEE, HTE
Professional Sponsors

General Notes

Please read this page carefully!

- There are six problems A, B, C, D, E, and F.
- To solve the problems, you can use any platform, operating system, programming language, development tools, libraries, etc. but you must not use any external human resources.
- With the exception of problem A, the input file is always a text file. The lines are separated by a single line feed character (character code 10), the last line of the file is also terminated by an LF character. The output of your programs also has to follow this convention. Moreover, there cannot be trailing space characters at the end of the lines.
- There are 10 test cases for each problem. If not written otherwise, then the input files for problem X are called `X0.in`, `X1.in`, . . . , `X9.in` where X is one of A, B, C, D, E, and F. The output files produced by your program should be called `X0.out`, `X1.out`, etc.
- With the exception of problem C, we check your solution by comparing your submitted outputs to our reference outputs. We accept an output file for a test case if it is *exactly* the same as ours. We do a simple binary comparison and we do not tolerate even a single bit of difference. Therefore please follow exactly our specifications with respect to spaces, new line characters, upper case/lower case, etc. The problems were formulated in such a way that there is only a unique correct solution. In problem C, we check your output pixel by pixel, and accept your solution only if it is exactly the same image as the reference output.
- We have provided some sample input/output files for the problems. The first sample input file for Problem X is called `sample-X-1.in`, a correct solution for this test case is given in the file `sample-X-1.out`. We suggest you to check your program with the sample files first, and submit your solution only if your output matches the sample outputs.
- To submit your solution, you should login at `http://www.challenge24.org/reg.php?cmd=login` using your team name and password. Then follow the *Electronic contest* link.
- You are required to submit exactly *one .zip file* called `X.zip` for each problem X. This .zip file should contain the output files for each test case of the given problem. Please note: using .zip compression is a must (we *do not* accept .tgz files)! Multiple uploads per problem is allowed, however, only the last .zip file is judged. The output files should be in the root directory of the .zip file, do not make any subdirectories.
- If your output is correct for a test case, then you will get 1 point. You get the point even if your solution for the other test cases are incorrect. Note that we check only your *last* submission for the problem. Therefore, if you submit a correct solution, and later you submit an incorrect solution for the test case, then you will get no points for this test case.
- The 30 teams scoring the highest points will gain the right to participate in the final at Budapest, Hungary April 29–May 1, 2005. In case of a tie, the sum of the submission times decides the order. *Submission time* of a problem is the time (in minutes) elapsed between the beginning of the contest (10 am GMT+1) and the last upload for this problem; or 0 if you received 0 point for the given problem. We always use server time.

- If you find a problem statement ambiguous, questions should be sent directly to the e-mail address `challenge24@challenge24.org`. You must not use `contest2005@challenge24.org`! Questions are answered privately.
- In the first hour of the contest, we also send replies immediately to `contest2005@challenge24.org`. Later a digest of the answers is sent in every 60 minutes. Please check the mailing list regularly so you don't miss any important announcement. It is highly recommended to subscribe to this list if you have not done it yet at http://www.challenge24.org:81/sympa/subrequest/contest2005
- If you have time, then please take photos during the contest. We would be glad to put any such photos on the contest website.

Good luck and have fun!

Problem F: Features

Introduction

The United Widget Factory is the leading producer of widgets in the country. Its most successful product is Widget X1, which had a 90% market share last year. To increase the profit of the company, the management decided to introduce the next generation of widgets. They hope that will persuade customers currently using gadgets to switch to widgets.

The R&D department proposed n new features that could be added to the new Widget X2. Of course, adding a new feature increases the cost of the product. After conducting a market survey, the Marketing Department realized that adding all the features would make the prize unrealistically high; therefore, only a subset of the n new features will be implemented in Widget X2. There are 2^n possible combinations, with the cheapest being implementing no new features at all. The Marketing Department do not really want to ponder the pros and cons of each and every combination: they just want a product that is cheap, but has some interesting new features. Therefore, they will just say that let's build the i-th cheapest combination, for some number i.

For example, assume that there are 3 new features, with costs 6, 1, 8, respectively. In this case we have the following possible combinations, ordered by increasing cost:

Rank	Features	Cost
1	No new features	0
2	2	1
3	1	6
4	1,2	7
5	3	8
6	2,3	9
7	1,3	14
8	1,2,3	15

Therefore, if the Marketing Department wants the 6th cheapest combination, then this means that features 2 and 3 will be implemented in the new version.

Input

Each input file contains several test cases. Each test case consist of several integer numbers on a single line. The first number is $1 \le n \le 30$, the number of proposed features. The second number $1 \le i \le 2^n$ is the number given by the Marketing Department. These two numbers are followed by n integers: the cost of the n new features. The cost of each feature is between 1 and 10000.

The input is terminated by a test case with $n = i = 0$.

Output

For each test case, you have to output a single integer number on a separate line: the cost of the i-th cheapest combination.

Note: It is possible that there are multiple combinations having the same cost. For example, if there are 3 features with costs 10, 20, and 30, then we have the following combinations:

Rank	Features	Cost
1	No new features	0
2	1	10
3	2	20
4	1,2	30
5	3	30
6	1,3	40
7	2,3	50
8	1,2,3	60

Since all you have to output is the cost of the *i*-th combination, the result does not depend on the order in which the combinations having the same cost are listed. In the example above, changing the order of combination 1,2 and combination 3 would not change the result.

Sample Input

```
3 1 10 20 30
3 2 10 20 30
3 3 10 20 30
3 4 10 20 30
3 5 10 20 30
3 6 10 20 30
3 7 10 20 30
3 8 10 20 30
0 0
```

Sample Output

```
0
10
20
30
30
40
50
60
```

Problem E: Easy Compression

Introduction

Intergalactic sub-ether communication is very expensive: you have to pay $6 \cdot 10^{23}$ credits per character. Therefore, it is essential to minimize the length of the messages. You have decided to use a very simple compression algorithm: you find out what the most frequent word in the text is, and you replace every occurrence of this word with the character `'*'`.

Input

The input file is a text file, each line consists of a series of words separated by one or more space characters. A word can be any sequence of characters, it can contain symbols such as `','` or `':'` etc.

Output

The output should be the same text file as the input. The only change that you have to make is to replace the most frequent word with the character `'*'`. It can be assumed that there is a unique word that appears the highest number of times. Line breaks, leading and trailing spaces, etc. should appear in the output exactly the same way as it appeared in the input.

Sample Input

```
four one  three three four
 four  three
four
```

Sample Output

```
* one  three three *
 *  three
*
```

Problem D: Destruction Droid

Introduction

Life was easy and happy on the third moon of Sirius 8: people lived and worked in the many peaceful colonies. However, the Klingon Empire decided to extend its control to this part of the galaxy: they landed a Destruction Droid^{TM} on the surface of the moon. The Destruction Droid^{TM} will go to one colony and destroy it completely. If we knew which colony is targeted by the Destruction Droid^{TM}, then we could concentrate our forces to defend that colony. Fortunately, we have intercepted the commands transmitted to the Destruction Droid^{TM}, thus it is possible to determine its destination. Your task is to write a program that determines the unfortunate colony where the Destruction Droid^{TM} goes.

One of the nice things about this moon is that it is almost completely flat, hence everything can be described easily with two coordinates. East is x direction, North is y direction.

Input

Each input file begins with a line containing four integers:

- $n \leq 500$, the number of commands,
- $m \leq 100$, the number of colonies,
- x and y, the starting coordinates of the Destruction Droid^{TM}.

The first line is followed by n lines containing one command each. There are 5 different commands:

- LEFT p: turn left p degrees
- RIGHT p: turn right p degrees
- FASTER p: increase the speed with p units
- SLOWER p: decrease the speed with p units
- WAIT p: the Destruction Droid^{TM} goes for p time units in the current direction with the current speed

Speed is given in coordinate units per time units. It can be assumed that the commands do not increase the speed above 500 or below 0. After landing, the Destruction Droid^{TM} faces North (y direction), and its speed is 0. The commands LEFT, RIGHT, FASTER, and SLOWER are executed in zero time: the Destruction Droid^{TM} does not move during these commands.

The last m lines of the input describe the colonies. Each line begins with a name of length at most 20, which does not contain any space characters. The name is followed by the two coordinates of the colony.

Output

You have to output the name of the colony where the Destruction Droid^{TM} will be after executing the commands. Write a new line character after the name of the colony. It can be assumed that the Destruction Droid^{TM} will be at a distance of at most 10 from some colony, and there is a unique such colony. It is possible that the Destruction Droid^{TM} goes through some other colonies while executing the commands, but we do not care about that.

Sample Input

```
5 4 0 0
FASTER 10
WAIT 10
RIGHT 135
SLOWER 5
WAIT 28
Aaaa 0 0
Bbbb 100 100
Cccc 0 100
Dddd 100 0
```

Sample Output

```
Dddd
```

Problem C: Cleaning the Image

Introduction

A space probe has landed on Pluto and made some very interesting scientific measurements. The probe transmitted back to Earth all the data, including some high resolution pictures. However, the pictures were transmitted in analog format, hence they got more and more blurred every time they were transmitted, received, and relayed. Your job is to write a program that recovers the original images.

Each original image is an $n \times n$ matrix of black and white pixels: 0 represents black and 1 represents white. If we analyze the way the image reached the Earth, it turns out that the image was blurred exactly 5 times. Each time the image was blurred, the value of every pixel was increased by the sum of its 8 neighbors. This means that every pixel is between 0 and 9 after blurring the image for the first time; every pixel is between 0 and 81 after blurring it once more, etc. The boundaries of the image is handled in a wrap-around fashion: if a pixel is on the left edge of the image, then it has neighbors on the right edge, etc. For example, blurring the 4×4 image on the left gives the second matrix; blurring it once more gives the matrix on the right:

```
1 0 0 0    1 1 0 1    13 10 12 10
0 0 0 0    1 2 1 2    10 8  10 10
0 0 1 0    0 1 1 1    12 10 13 10
0 0 0 0    1 2 1 2    10 8  10 8
```

Input

For each test case, the input is a binary file describing the blurred image as it was received (recall that the image was blurred 5 times). The size of every image is 256×256. Each pixel is given by a 32 bit integer, stored in 4 bytes in LSB first format. Hence the size of each file is $256 \times 256 \times 4 = 262144$ bytes. The pixels are given in row major order: the first 256×4 bytes describe the first row, etc.

Output

For each test case you have to output a `.gif` file containing the recovered image. The size of the image should be 256×256, and it should contain only black (RGB(0,0,0)) and white (RGB(255,255,255)) pixels.

Problem B: Barbarian Invasion

Introduction

The peaceful kingdom of Bandulu is threatened by the invading barbarians! Each city in the kingdom has some number of guards, but they are no match for the blood-thirsty barbarian hordes. However, if all the guards are protecting the same city, then they can save one city. Therefore, King Dulu III decided that all the guards should move to one city, which will be transformed into an unconquerable stronghold. All the other cities will be protected by one huge wall that encloses all these cities. Building walls is very expensive, hence it is important to build the shortest possible wall. Therefore, it should be selected very carefully which city will be the stronghold. Your task is to write a program that determines the minimum length of the wall that should be built.

There is another danger that has to be taken into account when building the walls. The cities communicate each other with a communication network: there is an optical cable between each pair of cities. This cable goes on a straight line between the two cities. The safety of the communication network is absolutely necessary for the everyday life of the country. Therefore, it is not enough to protect the cities with the walls, the walls should protect the cables as well. The walls should be built in such a way that no cable goes on the outside of the walls. The only exception is the cables going to the stronghold: these cables have to go outside the walls anyway, so we do not care about them. If there is a segment of the wall that connects city A and city B, then we assume that the cable connecting A and B is protected by this wall.

Input

The input contains several blocks of test cases. Each case begins with a line containing a single integer $1 \le n \le 2000$, the number of cities. The next n lines contains two numbers each: the coordinates of the cities. Each coordinate is between 0 and 20000.

The input is terminated by a block with $n = 0$.

Output

For each test case, you have to output a line containing a single integer, the minimum length of the wall (rounded down).

Sample Input

```
5
0 0
10 0
10 10
0 10
100 0
4
0 0
100 0
0 100
11 9
0
```

Sample Output

```
40
203
```

Problem A: Archipelago

We did not have enough time to write a description for this problem. Sorry about that. Please use the sample input and output files to figure out what the task is, then generate all the output files for the test cases.