

HKUST Programming Contest 2008 Spring*

Date: April 27th, 2008

Time: 13:00 – 17:30(17:50)

Venue: CS Lab 3

***Acknowledgement: problems are from TopCoder & ICPC 2006 Japan**

Rules:

1. Teams are ranked according to the most problems solved. Teams who solve the same number of problems are ranked by least total time. The total time is the sum of the time consumed for each problem solved. The time consumed for a solved problem is the time elapsed from the beginning of the contest to the submission of the accepted run plus 20 penalty minutes for every rejected run for that problem regardless of submission time. There is no time consumed for a problem that is not solved.
2. The correctness of each submission is judged by inputting test cases into the submitted program. The submission is regarded as correct if its output matches completely with the judge's output. The submission is judged as correct or wrong. No partial credit is given.
3. The contestants can re-submit another source code after previous wrong submissions.
4. The contestants should read the input from standard input and print the output to standard output for all the problems.
5. Each problem may have different time limit.
6. **The programming languages to be used in this contest are C++VS2005, C++VS6.0 and JAVA(1.6.0). The contestants use PC² to submit their source codes to the judge and the source codes are compiled by your selection. Please don't make wrong choice and notice that VS2005 and VS6.0 compilers are different.**
7. The contestants are allowed to bring any hard copies of books, notes, references, dictionaries and sketch papers to the contest site. Electronic devices are forbidden.
8. This is an individual-based contest. Contestants should follow rules and no cheating/collaboration behavior is allowed. Any offence of this rule will be punished according to the university academic regulations.
9. For your convenience, you can download sample input data from <http://www.cse.ust.hk/acm/contest/sample.zip>. But your program should read from standard input channel (*stdin*, *cin*, *etc.*).
10. Put your code at "i:\drive\homes". Reboot of your computer will remove all programs.

Problem A – Transformation

Time Limit: 5 seconds

You are to transform a positive integer vector (A_1, A_2, \dots, A_n) into another positive integer vector (B_1, B_2, \dots, B_n) of the same length. The transformation must satisfy the conditions below:

- 1) For $1 < i <= n$, B_i must evenly divide into A_i .
- 2) The least common multiple of all A_i should be equal to that of all B_i .

You are given the original integer vector A , where the i -th (1-based) element is a positive integer representing A_i . You need to calculate the transformed integer vector B . Each element of B must be a positive integer with no leading zeroes. If there are multiple solutions, return the one where the first number is the smallest. If there are still multiple solutions, return the one with the smallest second number etc.

Input

There are multiple test cases in the input. Each test case consists of one line. The line starts with a number n ($1 \leq n \leq 50$), following with n space separated positive integers indicating a vector A . Each element of A will be a positive number smaller than 10^{18} . The input ends with $n=0$.

Output

For each test case, write a single line containing the corresponding vector B . You should separate elements by 1 space. Do not print extra space at the end of the line.

Sample Input

```
2 1 2
3 2 3 6
5 2 10 2 3 5 7
5 6 2 3 4 9
6 6 2 3 4 9 8
8 3637 260 26122993443584 47715564111559878 2 871126696052836 3492829317 83024857214176826
0
```

Sample Output

```
1 2
1 1 6
1 2 3 5 7
1 1 1 4 9
1 1 1 1 9 8
3637 5 26122993443584 7952594018593313 1 217781674013209 3492829317 41512428607088413
```

Problem B – Olympic Games

Time Limit: 15 seconds

There are N teams taking part in The Programming Olympic Games. They are numbered by integers between 0 and $N-1$, inclusive. According to the rules, the Games consist of many competitions. The three top scoring teams within each competition are awarded gold, silver and bronze medals. Within a single competition, there can be no ties, and a single team can win at most one medal. After the Games are over, the following rules are used to rank the teams:

- If two teams each have a different number of gold medals, the team with more gold medals is ranked higher. Otherwise, the next rule is used.
- If two teams each have a different number of silver medals, the team with more silver medals is ranked higher. Otherwise, the next rule is used.
- If two teams each have a different number of bronze medals, the team with more bronze medals is ranked higher. Otherwise, the lower numbered team is ranked higher.

The Games are now in progress and *left* competitions are left before the Games are over. You are given an array *medals*, the i -th element of which represents the current number of medals won by the i -th team. Each element is formatted "GOLD SILVER BRONZE", where GOLD is the number of gold medals, SILVER is the number of silver medals, and BRONZE is the number of bronze medals.

Team 0 is very strong in the disciplines represented by all of the remaining competitions. Therefore, they will definitely win gold medals in all of them. Please compute the worst possible final ranking for team 0, where 1 is the highest ranking, 2 is the second highest, and so on.

Input

There will be multiple test cases (less than 150 test cases). Each test case starts with n ($3 \leq n \leq 50$) which is the number of teams in this test case and *left* ($0 \leq \textit{left} \leq 10000$) which is the number of competitions left. Next n lines each contain three integers for "GOLD, SILVER BRONZE" of a team. These n lines correspond to team 0 to team $n-1$.

The input ends with $n=0$ and *left* = 0.

Output

For each test case, output an integer on a single line indicating the worst possible ranking of team 0.

Sample Input

```
3 1
0 0 0
1 0 0
1 0 0
3 5
0 5 5
```

5 0 0
5 0 10
7 3
0 1 1
3 0 0
3 0 0
3 0 1
3 1 0
3 1 1
3 1 1
0 0

Sample Output

3
2
5

Explanation

In the first test case, in the only 1 match left, if team 0 wins gold, team 1 wins silver, team 2 wins bronze, team 0 will end up with 3rd place in the ranking.

In the second test case, team 2 can rank higher than team 0 if it gets 5 silvers.
In the 3rd test case, team 3 wins 2 silver medals, team 4 wins 2 bronze medals, team 5 wins 1 silver, and team 6 wins 1 bronze. Team 0 would rank 5th.

Problem C – MST Span Time Limit: 5 seconds

Given an undirected weighted graph G , you should find one of spanning trees specified as follows.

The graph G is an ordered pair (V, E) , where V is a set of vertices $\{v_1, v_2, \dots, v_n\}$ and E is a set of undirected edges $\{e_1, e_2, \dots, e_m\}$. Each edge e in E has its weight $w(e)$.

A spanning tree T is a tree (a connected subgraph without cycles) which connects all the n vertices with $n - 1$ edges. The slimness of a spanning tree T is defined as the difference between the largest weight and the smallest weight among the $n - 1$ edges of T .

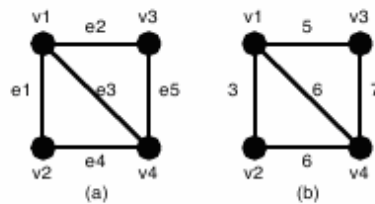


Figure 1

For example, a graph G in Figure 1(a) has four vertices $\{v_1, v_2, v_3, v_4\}$ and five undirected edges $\{e_1, e_2, e_3, e_4, e_5\}$. The weights of the edges are $w(e_1) = 3$, $w(e_2) = 5$, $w(e_3) = 6$, $w(e_4) = 6$, $w(e_5) = 7$ as shown in Figure 1(b).

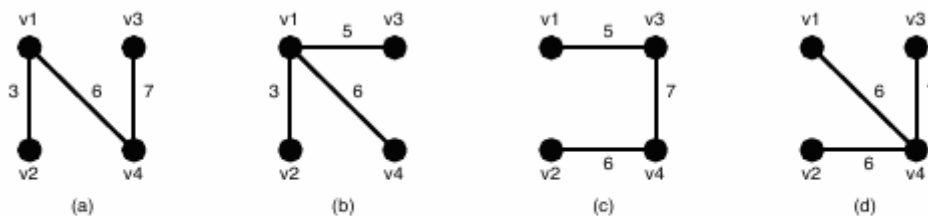


Figure 2

There are several spanning trees for G . Four of them are depicted in Figure 2(a)-(d). The spanning tree T_a in Figure 2(a) has three edges whose weights are 3, 6 and 7. The

largest weight is 7 and the smallest weight is 3 so that the slimness of the tree T_a is 4. The slimnesses of spanning trees T_b , T_c and T_d shown in Figure 2(b), (c) and (d) are 3, 2 and 1, respectively. You can easily see the slimness of any other spanning tree is greater than or equal to 1, thus the spanning tree T_d in Figure 2(d) is one of the slimmest spanning trees whose slimness is 1.

Your job is to write a program that computes the smallest slimness. Your algorithm should be efficient to meet the time limit requirement.

Input

The input consists of multiple datasets, followed by a line containing two zeros separated by a space. Each dataset has the following format.

```

n m
a1 b1 w1
⋮
am bm wm

```

Every input item in a dataset is a non-negative integer. Items in a line are separated by a space. n is the number of the vertices and m the number of the edges. You can assume $2 \leq n \leq 100$ and $0 \leq m \leq n(n-1)/2$. a_k and b_k ($k = 1, \dots, m$) are positive integers less than or equal to n , which represent the two vertices v_{a_k} and v_{b_k} connected by the k -th edge e_k . w_k is a positive integer less than or equal to 10000, which indicates the weight of e_k . You can assume that the graph $G = (V, E)$ is simple, that is, there are no self-loops (that connect the same vertex) nor parallel edges (that are two or more edges whose both ends are the same two vertices).

Output

For each dataset, if the graph has spanning trees, the smallest slimness among them should be printed. Otherwise, '-1' should be printed. An output should not contain extra characters.

Sample Input

```
4 5
```

1 2 3
1 3 5
1 4 6
2 4 6
3 4 7
4 6
1 2 10
1 3 100
1 4 90
2 3 20
2 4 80
3 4 40
2 1
1 2 1
3 0
0 0

Sample Output

1
20
0
-1

Problem D – Most Distant Point from the Sea

Time Limit: 5 seconds

There is an island surrounded by the sea. In such an island, it is natural to ask a question: "Where is the most distant point from the sea?"

In this problem, you are asked to write a program which, given a map of an island, finds the most distant point from the sea in the island, and reports its distance from the sea. In order to simplify the problem, we only consider maps representable by convex polygons.

Input

The input consists of multiple datasets. Each dataset represents a map of an island, which is a convex polygon. The format of a dataset is as follows.

N

$x_1 y_1$

...

$x_n y_n$

Every input item in a dataset is a non-negative integer. Two input items in a line are separated by a space.

n in the first line is the number of vertices of the polygon, satisfying $3 \leq n \leq 100$. Subsequent n lines are the x - and y -coordinates of the n vertices. Line segments $(x_i, y_i) - (x_{i+1}, y_{i+1})$ ($1 \leq i \leq n - 1$) and the line segment $(x_n, y_n) - (x_1, y_1)$ form the border of the polygon in counterclockwise order. That is, these line segments see the inside of the polygon in the left of their directions. All coordinate values are between 0 and 10000, inclusive.

You can assume that the polygon is simple, that is, its border never crosses or touches itself. As stated above, the given polygon is always a convex one.

The last dataset is followed by a line containing a single zero.

Output

For each dataset in the input, one line containing the distance of the most distant point from the sea should be output. An output line should not contain extra characters such as spaces.

The answer should not have an error greater than 0.00001 (10^{-5}). You should round your result to 6 digits after the decimal point.

Sample Input

```
4
0 0
10000 0
10000 10000
0 10000
3
0 0
10000 0
7000 1000
6
0 40
100 20
250 40
250 70
100 90
0 70
3
0 0
10000 10000
5000 5001
0
```

Sample Output

```
5000.000000
494.233641
34.542948
0.353553
```

Problem E – Fair Dice

Time Limit: 30 seconds

BattleDice is a two-player game where each person uses his own N -sided die. The two players roll their dice, and whoever's die shows the highest number wins. In the event of a tie, they continue rolling until one player wins. You and your opponent start with a total of 3 dice. Your opponent selects one of the three, and then you select one of the remaining two.

To make the game more interesting, you and your opponent begin with only two of the three dice. Given these two, determine the third die you would like to construct and add to the original two in order to maximize your probability of winning the game. Assume that your opponent will play optimally, making the selection that will give him the highest chance of winning.

The two dice will be given as two vectors $die1$ and $die2$. The numbers on each side of the dice will be between 1 and $range$, inclusive, and the numbers on the third die you are constructing must also be in this range. The two given dice will have the same number of sides, and the die you are constructing must have the same number of sides as the first two. You need to return your die as an array, sorted in ascending order.

Neither of the two given dice will have the same number on every side, but your third die may.

There may be more than one third die that will give you the same optimal chance of winning the game. In this case, sort the values on each die in ascending order, find the first pair of corresponding values that differ, and prefer the die with the smaller of those two values.

For example, if $range = 6$ and the first two dice are:

$$\begin{aligned} die1 &= \{ 2, 2, 5 \} \\ die2 &= \{ 2, 3, 3 \} \end{aligned}$$

you would choose to construct:

$$die3 = \{ 1, 3, 4 \}$$

If your opponent selects $die1$, you could choose $die2$ to give yourself a $4/7$ chance of winning. If your opponent selects $die2$, you could choose $die3$ to give yourself a $4/7$ chance of winning. If your opponent selects $die3$, you could choose $die1$ to give yourself a $5/9$ chance of winning. Given these three choices your opponent will select $die3$, and you will have a $5/9$ chance of winning. No other third die will give you a higher chance of winning. The die $\{ 1, 4, 4 \}$ also guarantees you a $5/9$ chance of winning, but according to the tiebreaking rules $\{ 1, 3, 4 \}$ is the correct answer.

The dice used are fair dice, which means that every roll will result in each side with a $1/N$ probability.

Input

There would be multiple test cases. Each test case starts with two integers n ($2 \leq n \leq 10$) and *range* ($2 \leq \text{range} \leq 15$). Then two lines each contain n integers, indicating number on the faces of die1 and die2, respectively. The last line of input contains two zeros.

Output

For each test case, output one line containing n integers for the corresponding test case, indicating the 3rd die you want to construct. The n integers should be in the range of 1 and *range*, in ascending order. They should be separated by single space and you have to output the lexically smallest answer.

Sample Input

```
3 6
2 2 5
2 3 3
4 12
1 3 8 10
1 5 6 12
7 15
6 6 8 9 13 13 15
3 5 8 10 13 14 14
0 0
```

Sample Output

```
1 3 4
1 6 6 7
1 7 9 10 13 13 14
```

Problem F – Prime Gap

Time Limit: 5 seconds

The sequence of $n - 1$ consecutive composite numbers (positive integers that are not prime and not equal to 1) lying between two successive prime numbers p and $p + n$ is called a prime gap of length n . For example, $\langle 24, 25, 26, 27, 28 \rangle$ between 23 and 29 is a prime gap of length 6.

Your mission is to write a program to calculate, for a given positive integer k , the length of the prime gap that contains k . For convenience, the length is considered 0 in case no prime gap contains k .

Input

The input is a sequence of lines each of which contains a single positive integer. Each positive integer is greater than 1 and less than or equal to the 100000th prime number. The end of the input is indicated by a line containing a single zero.

Output

The output should be composed of lines each of which contains a single non-negative integer. It is the length of the prime gap that contains the corresponding positive integer in the input if it is a composite number, or '0' otherwise. No other characters should occur in the output.

Sample Input

```
10
11
27
2
492170
0
```

Sample Output

```
4
0
6
0
114
```