Colombian Collegiate Programming League
CCPL 2020

Round 1 – February 22

Problems

This set contains 11 problems; pages 1 to 18.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

A - A Contest to Meet

Source file name: `acm.c`, `acm.cpp`, `acm.java`, *or* `acm.py`

A Contest to Meet (ACM) is a reality TV contest that sets three contestants at three random city intersections. In order to win, the three contestants need all to meet at any intersection of the city as fast as possible. The contestants are given communication devices to help them in sharing information about the city (e.g., where the sun is, how far the mountains are, etc.).

It should be clear that the contestants may arrive at the intersections at different times, in which case, the first to arrive can wait until the others arrive. Moreover, it is guaranteed that there is a path between any pair of intersections.

From an estimated walking speed for each one of the three contestants, ACM wants to determine the minimum time that a live TV broadcast should last to cover their journey regardless of the contestants' initial positions and the intersection they finally meet. You are hired to help ACM answer this question.

You may assume the following:

- Each contestant walks at a given estimated speed.
- The city is a collection of intersections in which some pairs are connected by bidirectional streets that the contestants can use to traverse the city.

Input

The input consists of several test cases.

The first line of each test case contains two blank-separated integer values N and S, indicating the number of intersections and the number of streets in the city, respectively ($1 \leq N \leq 100$, $0 \leq S \leq 9000$). Then follow S lines, each one with three blank-separated integer values i, j, and d ($0 \leq i < N$, $0 \leq j < N$, $i \neq j$, $1 \leq d \leq 200$), meaning that there is a street of length d meters connecting the i-th and the j-th intersections. Note that intersections are indexed from 0 to $N - 1$. You can assume that there is at most one street connecting any pair of intersections and that the input lists a street exactly once.

At the end of each test case there is one line with three blank-separated integer values sA, sB, sC ($50 \leq sA, sB, sC \leq 100$), representing the walking speed of each of the three contestants, in meters per minute.

The input must be read from standard input.

Output

For each test case, print a single line with an integer indicating the minimum number of minutes that will pass before the three contestants can meet, if they start to walk immediately after the show starts. Remember that the contestants can start at any random (unknown) intersection and can decide to meet at any intersection: you need to account for the worst case scenario.

The answer should be given rounding decimals to the next integer (e.g., 2.9 minutes rounds up to 3 minutes and 3.2 minutes rounds up to 4 minutes).

The output must be written to standard output.

Sample Input	Sample Output

```
2 1
0 1 150
60 50 75
3 2
1 0 100
1 2 80
60 80 50
4 5
0 1 200
0 2 200
0 3 200
2 1 200
2 3 200
50 100 100
```
	```
	3
	4
	8
	```

B - Kitchen Mess

Source file name: `kitchen.c`, `kitchen.cpp`, `kitchen.java`, *or* `kitchen.py`

Remy is an intelligent mouse with a highly developed sense of taste; he has become one of the most renowned chefs in Paris. However, his popularity has caused a lot of skepticism between human chefs and he has started to receive challenges to test his abilities.

Recently, Remy received a challenge from chef Gusteau to compete for dessert: the best tiramisu recipe. Even though Remy is a skilled chef, he has never focused on desserts. Moreover, he does not even have the right tools for deserts. He has asked Alfredo, a friend who owns a bakery, to lend him the bakery kitchen to try out recipes. Alfredo accepts this request under two conditions: the kitchen can only be used at night, and the next morning it should be completely clean and tidied up. This also means that all baking trays are required to be stacked up in the least possible number of piles. All baking trays are rectangular with various lengths and widths: a tray can only be stacked on top of another tray that has equal or larger dimensions.

One night, after trying out many times, Remy finally discovered the perfect tiramisu recipe. However, he was so tired (energy in mice can somehow be easily depleted) that after washing all bakery trays, night caught up with him and felt asleep. Next morning, he was waked up by the sounds of people coming to open the bakery. Quickly, Remy tried to put everything into place, but one thing was left: to pile up the baking trays according to the rule of the pastry chef.

Remy needs help: can you find the least possible number piles in which all the baking trays can be put complying with Alfredo's rule.

Input

The input consists of several test cases. Each test case begins with a line containing one integer n ($1 < n \leq 1\,000$), the number of baking trays available in the kitchen. Each of the following n lines describes the dimensions of each baking tray: two blank-separated integers x and y indicating the width and length of the tray ($1 \leq x \leq 1\,000$, $1 \leq y \leq 1\,000$).

The input must be read from standard input.

Output

For each test case, output a single line with the least number of piles that can be formed using all the baking trays under the given condition.

The output must be written to standard output.

Sample Input	Sample Output

3
3 3
2 1
1 2
3
3 3
2 2
9 1
	1
	2

C - Computerabytech Inc.

Source file name: `computer.c`, `computer.cpp`, `computer.java`, *or* `computer.py`

Computerabytech Inc. is a technology company specialized in the development of string algorithms. The company staff is organized in teams, each one responsible for distinct projects. Doctor Terabyte, the CEO of the company, has hired you because of your unique abilities as stringologist to lead one of the teams. As you can see, the name of the company is formed joining together the words `TERABYTE`, `TECH`, and `COMPUTER`, in such a way that the resulting name is the shortest common string containing as substrings each one of the given words:

```
COMPUTERABYTECH
```

Each team in the company must have a name built following the same rule, using a set of words given by Doctor Terabyte. Of course, you do not want to test manually all options to find the team's name. As a proud stringologist, you are challenged to write a program to find the minimum length possible of a string containing as substrings each one of the strings in a given set.

Input

The input consists of several test cases. The first line of a test case contains a positive integer N indicating the number of words present in the set given by Doctor Terabyte ($2 \leq N \leq 16$). Then follow N lines, each containing one single word w that consists of uppercase letters from the English alphabet ($1 \leq |w| \leq 32\,768$). You may assume that the given set does not contain repeated words.

The input must be read from standard input.

Output

For each test case, output a single line with one integer indicating the minimum length attained by a string containing as substrings each one of the strings in the set given by Doctor Terabyte.

The output must be written to standard output.

Sample Input

```
3
TERABYTE
TECH
COMPUTER
3
COMPUTER
COMPANY
TECHNOLOGY
5
STRING
RING
INGENIOUS
OUSTER
FIRST
2
EARTH
ART
```

Sample Output

```
15
25
18
5
```

D - Ghost Hunting

Source file name: `hunting.c`, `hunting.cpp`, `hunting.java`, *or* `hunting.py`

A ghost Pokemon gang has been playing pranks on citizens and damaging public property in Lavender City. Any attempt to apprehend the gang has been futile, since it is impossible to see a ghost with the naked eye.

The major of Lavender City requested help from Silph Corp. After several months of research, the corporation developed a system that allows people to see the ghost Pokemon. The system is based on beacons: when a ghost Pokemon is surrounded by them, it is visible to anyone. However, the beacons use a lot of energy and thus require to be installed in light poles already available in the city. A Pokemon is *surrounded by beacons* if, in order to leave town, it needs to cross through a virtual fence: the idea is that any pair of beacons induce a virtual fence defined by the segment of a straight line starting at one of them and ending at the other one.

The past week, Silph Corp sent a truck full of beacons to create a visibility area in Lavender City. However, the ghost gang attacked the truck and damaged most of the beacons, except for three that are still fully operational. Since the production of beacons takes time, and the situation is getting dangerous by the day, the major wants to find the largest possible area that can be set up to see the ghosts once the three beacons are installed.

Given the location of the light poles, your help has been requested to find such an area.

Input

The input consists of several test cases. Each test case begins with a line containing an integer N ($3 \leq N \leq 2\times10^3$) representing the number of light poles in Lavender City. Each of the following N lines contains two blank-separated integers x and y ($-10^6 < x, y < 10^6$) indicating the location (x, y) of a light pole. The test cases end with the end of the input.

The input must be read from standard input.

Output

For each test case, output a single line with the largest possible area, truncated to one decimal, where the ghosts can be seen.

The output must be written to standard output.

Sample Input	Sample Output

4
-1 -1
-1 1
1 -1
1 1
3
-1 -1
-1 3
3 3
4
1 1
2 1
4 1
4 6
	2.0
	8.0
	7.5

E - Jawbreaking Candy

Source file name: `jawbreaking.c`, `jawbreaking.cpp`, `jawbreaking.java`, *or* `jawbreaking.py`

The *Advanced Cutting Machines* (*ACM*) has developed a new product for cutting rectangular candies into shorter pieces. The width of candies has been optimized already, so this machine's purpose is about optimizing the length of cuts. ACM is very excited about the new machine because it will solve the eternal discussion of how long candies should be for a given audience.

The in-house Mathematics Department of ACM determined how the machine cuts the pieces of candy. The lengths of candy that the machine can cut are those shorter than original one L and can be represented as a fraction of the form $\frac{a \cdot L}{b}$, where integers a and b have to satisfy $0 < a$ and $0 < b \leq n$. Here n represents a cutting resolution for the different models of machine that will be produced; more expensive models will have higher cutting resolution. For example, assume Alice wishes to buy a piece of candy of length at least 320 units. If this piece were to be cut from a longer piece of 500 units of length and the cutting machine can only cut fractions of the candy with a denominator at most $b = 3$, then the following lengths of candy could be cut:

$$\frac{500}{3}, \frac{500}{2}, \frac{500}{1}, \frac{1000}{3}, \frac{1000}{2}, \frac{1000}{1}, \frac{1500}{3}, \frac{1500}{2}, \frac{1500}{1}, \ldots$$

The in-house Mathematics Department predicts that, from these options, Alice would prefer the one with length $\frac{1000}{3}$ because it is the smallest one that is at least 320, as she wished, and will have less calories.

Given L, n, and a desired length of candy to cut d, compute the shortest candy length that the machine can cut that is at least d units of length presented in the form of a reduced fraction (i.e., the numerator and denominator cannot share positive factors other than 1).

Input

The input consists of several test cases, one per line. Each line contains integers $0 < L < 10^6$, $0 < n \leq 5\,000$, and $0 < d \leq L$ separated by a single space. The input ends with a line containing three blank-separated zeroes.

The input must be read from standard input.

Output

For each test case, output the shortest length that can be cut by the machine and that is at least d units of length in the form of a reduced fraction.

The output must be written to standard output.

Sample Input	Sample Output

`500 3 320`
`100 5 23`
`0 0 0`
	`1000/3`
	`25/1`

F - Grade Estimate

Source file name: `estimate.c`, `estimate.cpp`, `estimate.java`, *or* `estimate.py`

In his first programming course homework, Alex was given the following challenging task: given a positive integer n, he has been asked to design and implement an algorithm to find three integer numbers p, q, r such that $n = p \cdot q \cdot r$ and to print the value of $\frac{1}{n}$.

By now, Alex has submitted his solution to the homework. However, he has realized that in order to compute the value of $\frac{1}{n}$, his algorithm uses the following equality:

$$\frac{1}{n} = \frac{1}{p} + \frac{1}{q} + \frac{1}{r}.$$

This is indeed a problem because his algorithm works correctly depending on the choice of n. For example, if $n = 6$ the algorithm finds $p = 1$, $q = -2$ and, $r = -3$, and computes

$$\frac{1}{6} = \frac{1}{1} + \frac{1}{(-2)} + \frac{1}{(-3)}.$$

The final grade for this homework will be given as an integer number in the range 0..100: it corresponds to the percentage of cases correctly answered by each submission. Given the input that will be used to grade the homework, can you help Alex to estimate the grade of his submission?

Input

The input consists of several test cases. A test case is described by a line with m ($0 < m \leq 25$) positive integer numbers representing the set of numbers for evaluating Alex's task. Each number n in this list satisfies $0 < n \leq 82\,945\,319\,184$.

The input ends with a line containing the character 0.

The input must be read from standard input.

Output

For each test case, output one line containing an integer number corresponding to the integral part of the percentage of the m cases correctly answered by Alex's algorithm.

The output must be written to standard output.

Sample Input	Sample Output

5 6 7
42 6 3
10 20 1 1 3
6 42 82945319184
0
	33
	66
	0
	100

G - Rotating Drum

Source file name: `drum.c`, `drum.cpp`, `drum.java`, *or* `drum.py`

A rock 'n' roll band has k musicians, any of them can play any of n instruments, and they can be located in any order on the stage. This band has decided to make a drawing on the bass drum in order to characterize the way they perform on stage. The idea is to divide the surface of the bass drum into m equal sections (like a large pizza) and then assign one of k colors to each of the sections in a way that any possible sequence on n colors is found exactly once clockwise on the drum.

Nick De Bruijn – a musician in the band – is a mathematician and he knows that every possible sequence of n colors must be present on the bass drum. He knows that for $k \geq 2$ the value of m must be equal to k^n and for $k = 1$ the value of m must be equal to n.

As an example, consider the following bass drum drawing satisfying the abovementioned constraints for $k = 2$ and $n = 3$:

$$AAABABBB$$

In this case, each one of the 8 sequences appears exactly once clockwise in the drawing. Namely, the sequences AAA, AAB, ABA, BAB, ABB, BBB, BBA, BAA.

Your task is to help the band to find the sequence of colors that should be drawn on the bass drum for given k and n.

Input

The input consists of several test cases. Each test case is described by a line containing two blank-separated integers k and n: the number of colors ($1 \leq k \leq 26$) and the length of the subsequences ($1 \leq n \leq 10$). You may assume that $1 \leq m \leq 10^5$.

The input must be read from standard input.

Output

For each test case print a single line with the solution sequence. The k colors shall be represented by the first k uppercase letters of the English alphabet. If there is more than one solution, you must print the first sequence in lexicographical order.

The output must be written to standard output.

Sample Input	Sample Output
`4 2`	`AABACADBBCBDCCDD`
`2 3`	`AAABABBB`
`1 5`	`AAAAA`

H - Forming Better Groups

Source file name: `better.c`, `better.cpp`, `better.java`, *or* `better.py`

Prof. Smith is tired of the same old end of semester story: lazy students passing his course thanks to group assignments. Unfortunately, group assignments are an important part of the university policies and Prof. Smith cannot get ride of them. However, he has figured out a strategy that can limit the inclusion of lazy students as members of groups with hardworking students. He wants the lazy students to either work or fail his course.

The crux of the idea is to define a *threshold D* so that the following condition limiting how students participate in groups of *three* members is satisfied:

> if G_1, G_2, and G_3 are the grades obtained in the previous assignment by three students willing to form a group, then
>
> $$\max(G_1, G_2, G_3) - \min(G_1, G_2, G_3) \leq D.$$

Prof. Smith wants students to have many options to form groups and choose their teammates. Therefore, he would like to know in advance the number of ways his students could form groups of three members given threshold D and their grades in the previous assignment, while satisfying the condition above.

Please write a program to help Prof. Smith.

Input

The input has several test cases. The first line of a test case consists of two blank-separated integers N ($3 \leq N \leq 21$) and D ($0 \leq D \leq 500$), representing the number of students and the threshold, respectively. You can assume that N is a multiple of 3. The second line consists of a blank-separated list of grades $G_1, G_2, \ldots, G_N$ ($0 \leq G_i \leq 500$, for $1 \leq i \leq N$) corresponding to the grades of the N students in the previous assignment.

The input ends with $N = D = 0$.

The input must be read from standard input.

Output

For each test case, output one line containing one integer: the number of ways the N students can form groups of three members given the threshold D and the grades $G_1, G_2, \ldots, G_N$ in their previous assignment when the above-mentioned condition is enforced.

The output must be written to standard output.

Sample Input	**Sample Output**

`6 3`
`1 6 3 2 4 5`
`6 8`
`1 3 3 3 6 5`
`9 2`
`1 2 3 4 5 6 7 8 10`
`0 0`
	`2`
	`10`
	`0`

I - Beaded Bracelets

Source file name: `bracelets.c`, `bracelets.cpp`, `bracelets.java`, *or* `bracelets.py`

Beaded bracelets are a common kids craft. Building them can improve fine motor skills with the smaller movements that occur in the wrists, hands, and fingers. On top of that, once finished, the beaded bracelets can be worn as colorful inexpensive jewelry.

Once the glamorous craft of designing beaded bracelets is mastered, the next step is to align two of them into a single two-strips beaded bracelet. Since you are still new to this craft, only beads of two colors will be considered: blue (represented by '`B`') and red (represented by '`R`'). In this setting, the following are descriptions of two bracelets, each with four beads:

```
RBBB          BRBR
```

Since bracelets are circular, the above descriptions are not unique. If these two beaded bracelets were to be aligned (e.g., the first one on top of the second one), then the following two-strips bracelets could be built:

```
RBBB          RBBB
BRBR          RBRB
```

However, kids these days tend to be superstitious: a two-strips beaded bracelet in which two red beads are vertically aligned is *not acceptable* (otherwise, it is *acceptable*). In the two configurations above, the one on the left is acceptable, but the second one is not acceptable because the beads in the first row are both red.

This is the actual challenge for you: given the description of two bracelets made of blue and red beads only, how many rotations of the second bracelet result in an acceptable two-strips bracelet? Note that the two bracelets must be always aligned.

Input

The input consists of several test cases. A test case comprises three lines. The first line contains an integer n ($1 \leq n \leq 100\,000$) defining the number of beads in both bracelets. The next two lines each contains the description of a blue-red beaded bracelet as a sequence made of '`B`' and '`R`' characters of size n.

The input must be read from standard input.

Output

For each test case output the number of rotations of the second bracelet that result in an acceptable two-strips bracelet when the first bracelet is aligned on top of the second bracelet.

The output must be written to standard output.

Sample Input	**Sample Output**
4 RBBB BRBR 1 R B 2 RR BR	2 1 0

J - Fibcod Cryptocurrency

Source file name: `fibcod.c`, `fibcod.cpp`, `fibcod.java`, *or* `fibcod.py`

The Fibcod Cryptocurrency is the most popular payment method nowadays among participants of programming contests. To use it, participants are given a unique wallet number to send and receive money anywhere in the world, without paying any transfer fees!

As expected, assigned wallet numbers need to be unique. For this reason, Fibcod uses a huge collection of wallet numbers: they are large sequences made of the digits 0 and 1. However, unlike binary numbers where the digit 1 indicates that a given power of 2 needs to be added, the digit 1 in a Fibcod wallet number indicates which numbers in the following (shifted) Fibonacci sequence need to be added:

$$F_0 = 1$$
$$F_1 = 2$$
$$F_n = F_{n-1} + F_{n-2}, \quad \text{for } n \geq 2.$$

For example, the wallet number 11101 interpreted in base-10 is $17 = 8 + 5 + 3 + 1$.

Recently, one of the Fibcod Cryptocurrency founders noticed that different wallet numbers, when converted to base-10, identify the same wallet. For instance, 17 is also represented by the wallet number 100101 (i.e., $17 = 13 + 3 + 1$). Realizing that this goes against Fibcod's motto 'One for one and zero for none' and is likely to be the source of ugly legal disputes in the future, the cryptocurrency founders have agreed –in a bold and unprecedented decision– to ban the assignment of a wallet number if its base-10 interpretation has been assigned before.

Your task is to design and implement an efficient algorithm to convert Fibcod's wallet numbers to base 10, so that it can later be integrated to the wallet number assignment verification system.

Input

The input consists of several test cases, each comprising a Fibcod wallet number described in two lines. The first line contains a number $1 \leq n \leq 1\,000$ indicating the number of blocks of repeated 1s and 0s that concatenated define the wallet number. The second line contains n integer numbers $b_1, b_2, \ldots, b_n$ indicating how many times a given digit should be repeated per block: a b_i with odd index i, indicates a number of repeated 1s; a b_i with even index i, indicates a number of repeated 0s (for odd i, $1 \leq b_i \leq 100$; for even i, $1 \leq b_i \leq 10^9$). For example, in this format, the Fibcod wallet number 11000011111 is the concatenation of 3 blocks, where $b_1 = 2$ (i.e., 11), $b_2 = 4$ (i.e., 0000), and $b_3 = 5$ (i.e., 11111).

The input ends with a line containing 0 (which should produce no output).

The input must be read from standard input.

Output

For each test case, output the base-10 value of the given Fibcod's wallet number modulo 524 288.

The output must be written to standard output.

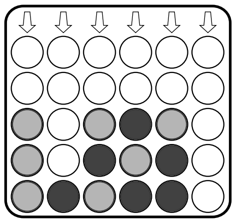
Sample Input	Sample Output

3
3 1 1
3
2 4 5
3
2 11 5
7
100 100000 100 100000 100 100000 100
0
	17
	252
	6784
	375403

K - Game Conundrum

Source file name: `game.c`, `game.cpp`, `game.java`, *or* `game.py`

Alice and Carl have a mean competitive siblings' rivalry. During summer, they play board games and declare an overall winner for the whole year. The summer holidays are about to end and they are tied with one final session to play in the family' favorite "4 In A Line". In this game, the two players have tokens of two different colors (Alice one color and Carl another color) and take turns to place them in a vertically set-up board of $n \times n$ size. Tokens are inserted from the top of a column and fall into the lowest unoccupied position in that column. The first player to complete a straight (horizontal, vertical, or diagonal) line of 4 tokens of its color wins the game.



At some point during the last session, you hear the hullabaloo of the summer. There is a big argument coming from the family's game room. When you arrive, you see the "4 In A Line" board on the floor and all the tokens played during the session scattered around. One of Alice and Carl, a sore loser, intentionally threw the board to the floor when the other player won the game. The thing is that both, Alice and Carl, are claiming the victory for the session and thus are the self-proclaimed overall winners of the year.

Given the number of tokens played by each one of Alice and Carl during the final session, and the name of who played the first token, you are to solve the game conundrum: who won the last session of the game?

Input

The first line of the input contains the number of test cases ($1 \leq g \leq 100$). Each test case consists of three lines describing a game. The first line of a game contains the side n of the board game ($4 \leq n \leq 128$). The second line contains two blank-separated integers a and c ($4 \leq a, c \leq 8\,192$) representing the number of tokens played by Alice and Carl, respectively. The third line contains the name of the player that started the game, either '`Alice`' '`Carl`'.

The input must be read from standard input.

Output

For each test case, output one line with the name of the winner of the last session: '`ALICE`' if Alice is the winner and '`CARL`' if Carl is the winner (ignore the quotes).

The output must be written to standard output.

Sample Input	**Sample Output**

`2`
`5`
`5 4`
`Alice`
`9`
`20 20`
`Carl`
	`ALICE`
	`ALICE`