НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**
Школа „Състезателно програмиране”
**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## A. Text Encryption

To keep privacy of messages and prevent the aliens from reading them, we may use various encryption algorithms. These algorithms encode a message into the so-called ciphertext that is difficult (or impossible) to decode for anyone else than the intended recipient. Transposition ciphers are a type of encryption that do not change the letters of the message but only change their order (“shuffle” the letters). Of course, the shuffling must be reversible to allow later decryption.
In this problem, we will consider a simple transposition cipher which shuffles the letters in such a way that the decryption algorithm always takes every n-th letter. More specifically: when decrypting, the first letter of the ciphertext is taken first, then the next n − 1 letters are (repeatedly) skipped and the next letter taken, and so on until we reach the end of the ciphertext. After that, we repeat the procedure starting with the second letter of the ciphertext, and so on until all letters are used.
Your task is to implement the encryption algorithm for this cipher. For a given message, produce the encrypted text (ciphertext). To make the cipher a little bit stronger, you should convert all letters to uppercase and leave out all spaces between words.

**Input Specification**
The input contains several messages. Each message is described by two lines. The first line contains one integer number N ($1 \le N \le 1000$). The second line contains the message. The message will be at most 10 000 characters long, it will only contain letters and spaces, and there will be at least one letter in each message. The last message is followed by a line containing zero.

**Output Specification**
For each message, output the ciphertext that, after using the described decryption algorithm, will result in the original message (with all spaces removed and all letters in uppercase).

**Sample Input**
```
2
CTU Open Programming Contest
7
This is a secret message that noone should ever see Lets encrypt it
15
text too short
0
```

**Output for Sample Input**
```
CMTMUIONPGECNOPNRTOEGSRTA
TESNUECHCAOLERIRGODLYSEENEEPITTEVTTSMHSESIAEAHRETSSTOSN
TEXTTOOSHORT
```

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**

Школа „Състезателно програмиране"

**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## B. Flip It!

Assume you have a set of cards laid out in an *n* by *m* grid. The cards are numbered and some are face up and others are face down. We can collapse the grid into a single pile by using a series of flips, each of which is one of the four following types:

- **Top Flip** : Here the cards in the top row are flipped over onto the corresponding cards on the row beneath them. Note that if a card is face up in the top row, it becomes face down after the flip, and vice versa. If the top row contains one or more piles of cards, each entire pile is flipped over like a stack of pancakes as it is moved to the lower row.
- **Bottom Flip** : Same as the Top Flip, but now the bottom row is flipped onto the next-to bottom row.
- **Left Flip** : Flip the cards in the left-most column onto the next-to-leftmost column.
- **Right Flip** : Flip the cards in the rightmost column onto the next-to-rightmost column.

After a series of n + m − 2 flips, the cards will be in a single pile, some cards face up and some face down. Your job is to determine the order of the face up cards in this final pile.

**Input**

Each test case will start with a line containing two positive integers n and m indicating the number of rows and columns in the grid. After this will come n rows of m integers indicating each card's number and its orientation. (The first row is the top row and the first value in each row is the leftmost card.) If a value is a positive integer k that means card k is at the location face up; if a value is a negative integer-k that means card k is at the location face down. (k will never be zero.) After these n rows there will be one more line containing n +m− 2 characters indicating the order of flips to apply to the grid. Each character will be either T, B, L or R corresponding to a top, bottom, left or right flip. All flip sequences will be legal, i.e., you won't be asked to do more than n − 1 top and bottom flips or m – 1 left and right flips. The maximum value for n and m is 20. A line containing two zeros will terminate input.

**Output**

For each test case, output the case number followed by a list of the numbers of all of the face up cards in the final deck, starting from the bottom of the deck. Follow the format used in the examples.

| Sample Input | Sample Output |
|---|---|
| 2 3<br>4 -17 -8<br>6 23 -5<br>LRB<br>1 1<br>-3<br><br>1 1<br>3<br><br>0 0 | Case 1: 8 6<br>Case 2:<br>Case 3: 3 |

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**
Школа „Състезателно програмиране”
**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## C. Numbersrebmun

Anna and Bob are starting up a new high-tech company. Of course, one of their key considerations is choosing a good name for the company. Palindromes are cool. (A palindrome is a word that is the same when reversed, like the names of our two entrepreneurs.) They would really the name of their company to be a palindrome. Unfortunately, they cannot think of a nifty company name that is also a palindrome.

Maybe at least the telephone number of their company could be a palindrome. However, they really want their customers to be able to call them, so they want to choose the company name so that, when it is typed using the letters printed on a phone keypad, the result is also their phone number. (On a standard phone keypad, the following keys contain the corresponding letters: 2: ABC, 3: DEF, 4: GHI, 5: JKL, 6: MNO, 7: PQRS, 8: TUV, 9: WXYZ.)

**Input Specification**

The first line of input contains a single integer, the number of lines to follow. Each following line contains a company name, which is a string of at most 20 letters, which may be either uppercase or lowercase.

**Sample Input**

```
2
ANBOBNA
iAmACoolCompany
```

**Output Specification**

For each company name, print a single line of output, containing the word YES if the phone number is a palindrome, or NO if it is not.

**Output for Sample Input**

```
YES
NO
```

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**

Школа „Състезателно програмиране”

**СЪСТЕЗАНИЕ, 13 април 2013 г.**

# D. ПОСЛЕДОВАТЕЛНОСТ

Да разгледаме редицата с първи елемент 1 и всеки следващ, получаващ се като се запише поредния му номер в редицата и два пъти се допише предходния елемент. Първите четири елемента на редицата изглеждат по следния начин:

1. 1
2. 211
3. 3211211
4. 432112113211211

Напишете програма, която по зададени $N$ и $K$ да намира $K$-тата цифра в записа на $N$-тия елемент на редицата (разбира се, ако има такъв).

**Вход**

Програмата трябва да може да обработва множество тестови примери. За поредният тест на отделен ред на стандартния вход се задават естествените числа N и K ($1 \le N \le 100000$, $1 \le K \le 10^{15}$). Края на входа е маркиран с 0.

**Изход**

За всеки тестов пример на отделен ред на стандартния изход програмата трябва да изведе търсената цифра. Ако дължината на N-тия елемент е по-малка от K, да се изведе -1.

| Вход | Изход |
|---|---|
| 5 4<br>2 25<br>0 | 2<br>-1 |

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**

Школа „Състезателно програмиране”

**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## E. Watering

Ели обича цветята, но често я мързи да полива тези на терасата. Вместо това тя разчита на дъжда да прави това вместо нея. Когато вали, Ели следи внимателно падането на капките и записва координатите на всяка от тях. Саксията с цветя на терасата е (доста) продълговата и за простота можем да я считаме за интервал с дължина **L** милиметра, а координатите на капките – число x между 0 и L, включително.

За „относително напоена” Ели счита саксията ако няма нейн подинтервал с дължина по-голяма от **D**, такъв че в него да не е паднала нито една капка. Когато Ели види, че саксията е станала относително напоена, тя може да спре да следи дъжда и да прави нещо друго.
Нека например саксията е с дължина 10, всеки ненапоен интервал трябва да е не по-дълъг от 4 милиметра, и целият дъжд се състои от 8 капки, паднали на координати {10, 2, 3, 8, 4, 2, 6, 5} (в този ред). След първите четири капки, ненапоените интервали са [0, 2), (2, 3), (3, 8), (8, 10). От тях интервалът (3, 8) е с дължина по-голяма от 4, следователно Ели не счита саксията за напоена. След падането на петата капката (с координати 4), обаче, той бива разцепен на (3, 4), (4, 8), като така вече никой от интервалите не е с дължина по-голяма от 4. Следователно Ели може да спре да следи дъжда след падането на 5-тата капка.

Напишете програма, която по зададена дължината на саксията **L**, максималното разстояние **D** и координатите на всички паднали капки по време на дъжда, определя след коя от тях Ели може да е спокойна, че саксията е относително напоена, или върнете **-1** ако дъждът не си е свършил работата и Ели трябва сама да се погрижи да напои цветята.

На първия ред на стандартния вход е зададен броят тестове **T**, които вашата програма трябва да обработи. Всеки от тях е описан на два реда. На първия от тях са зададени числата **N**, **L** и **D** – съответно броят капки, паднали по време на дъжда, дължината на саксията, и максималната дължина на ненапоен интервал. На следващия ред са зададени **N** числа между 0 и L, включително – координатите на капките в реда в който са паднали.

За всеки тест на отделен ред на стандартния изход изведете по едно число – минималния индекс на капка (броено от 1) след която Ели може да спре да гледа дъжда, или -1, ако дори след падането на всички капки саксията е все още неполята.

**Ограничения**

$1 \le \mathbf{T} \le 20$

$1 \le \mathbf{N} \le 100{,}000$

$1 \le \mathbf{D} < \mathbf{L} \le 1{,}000{,}000{,}000$

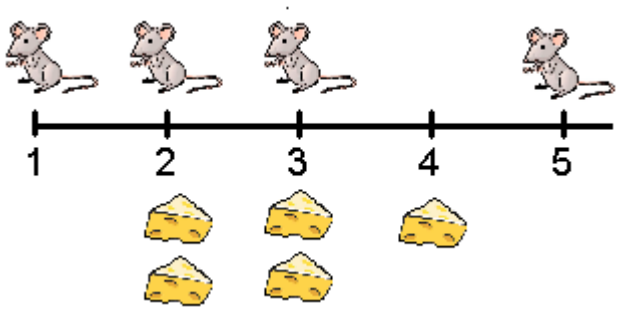
| *Примерен вход:* | *Примерен изход:* |
|---|---|
| 2<br>8 10 4<br>10 2 3 8 4 2 6 5<br>3 15 5<br>6 11 7 | 5<br>-1 |

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**
Школа „Състезателно програмиране”
**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## F. Мишки

*M* (1 ≤ M ≤ 100000) на брой мишки са разположени в една редица. Някъде между тях има парчета сирене. Всички мишки могат да се движат една след друга само наляво или само надясно и никоя мишка не задминава друга. За **всички** мишки има два варианта:




Вариант (А) - Всяка изяжда само едно парче сирене – първото неизядено парче, което срещне.

Вариант (Б) - Всяка изяжда всички неизядени парчета сирене, които срещне по пътя си.

На една позиция има най-много една мишка, но парчетата сирене може да са повече. Също така на една позиция може да има едновременно както мишка, така и парчета сирене. Напишете програма, която за всеки от двата варианта извежда посоката на движение и минималния брой мишки, неизяли нито едно парче сирене.

На първият ред на стандартния вход е записан броя на тестовете. На първия ред за всеки тест има цяло число *M* – броя на мишките. Следва ред съдържащ *M* на брой цели положителни числа: $m_1, m_2, m_3 \ldots m_m$ – позицията на всяка мишка в редицата. На третират ред е цяло число *N* (1 ≤ *N* ≤ 100000) – броя парчета сирене. Следва ред съдържащ *N* на брой цели положителни числа: $n_1, n_2, n_3 \ldots n_n$ – позицията на всяко парче сирене в редицата с мишки.

За всеки тест на първия ред на стандартния изход се отпечатва решението за вариант (А) - символ $P_1$ и число $B_1$, разделени с един интервал, където $P_1$ е посоката на движение на мишките, а $B_1$ е минималният брой мишки, които ще останат гладни. На втория ред на стандартния изход се отпечатва решението за вариант (Б) – символ $P_2$ и число $B_2$, разделени с един интервал, като $P_2$ и $B_2$ имат същите значения като $P_1$ и $B_1$. Стойностите на $P_1$ и $P_2$ са един от символите L, R или D, където: L е наляво, R е надясно, а D – минималният брой е един и същ при движение наляво или надясно.

| **Вход** | **Изход** |
|---|---|
| 2<br>6<br>2 3 8 9 11 12<br>5<br>1 4 6 7 10<br>4<br>1 2 3 5<br>5<br>2 2 3 3 4 | L 1<br>L 3<br>D 1<br>L 1 |

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**
Школа „Състезателно програмиране”
**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## G. Soda Surpler

Tim is an absolutely obsessive soda drinker, he simply cannot get enough. Most annoyingly though, he almost never has any money, so his only obvious legal way to obtain more soda is to take the money he gets when he recycles empty soda bottles to buy new ones. In addition to the empty bottles resulting from his own consumption he sometimes find empty bottles in the street. One day he was extra thirsty, so he actually drank sodas until he couldn't afford a new one.

## Input

The first line of the input file contains an integer N (N < 15) which denotes the total number of test cases. The description of each test case is given below:

Three non-negative integers *e,f,c,* where $e < 1000$ equals the number of empty soda bottles in Tim's possession at the start of the day, $f < 1000$ the number of empty soda bottles found during the day, and $1<c<2000$ the number of empty bottles required to buy a new soda.

## Output

For each test case print how many sodas did Tim drink on his extra thirsty day? Look at the sample output for details.

## Sample Input

```
2
9 0 3
5 5 2
```

## Sample Output

```
4
9
```

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**
Школа „Състезателно програмиране”
**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## H. Money Matters

Our sad tale begins with a tight clique of friends. Together they went on a trip to the picturesque country of Molvania. During their stay, various events which are too horrible to mention occurred. The net result was that the last evening of the trip ended with a momentous exchange of ``I never want to see you again!"s. A quick calculation tells you it may have been said almost 50 million times!

Back home in Scandinavia, our group of ex-friends realize that they haven't split the costs incurred during the trip evenly. Some people may be out several thousand crowns. Settling the debts turns out to be a bit more problematic than it ought to be, as many in the group no longer wish to speak to one another, and even less to give each other money.

Naturally, you want to help out, so you ask each person to tell you how much money she owes or is owed, and whom she is still friends with. Given this information, you're sure you can figure out if it's possible for everyone to get even, and with money only being given between persons who are still friends.

The first line of the input file contains an integer N ($N \leq 20$) which denotes the total number of test cases. The description of each test case is given below:

The first line contains two integers, $n$ ($2 \leq n \leq 10000$), and $m$ ($0 \leq m \leq 50000$), the number of friends and the number of remaining friendships. Then $n$ lines follow, each containing an integer $o$ ($-10000 \leq o \leq 10000$) indicating how much each person owes (or is owed if $o < 0$). The sum of these values is zero. After this comes $m$ lines giving the remaining friendships, each line containing two integers $x, y$ ($0 \leq x < y \leq n-1$) indicating that persons $x$ and $y$ are still friends.

For each test case your output should consist of a single line saying `POSSIBLE` or `IMPOSSIBLE`.

| Вход | Продължение на входа | Изход |
|---|---|---|
| 2<br>5 3<br>100<br>-75<br>-25<br>-42<br>42<br>0 1 | 1 2<br>3 4<br>4 2<br>15<br>20<br>-10<br>-25<br>0 2<br>1 3 | POSSIBLE<br>IMPOSSIBLE |

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**

Школа „Състезателно програмиране”

**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## I. Rain Fall

Rainfall is measured in millimeters. The rain is collected in a vertical transparent tube with millimeter markings, and once the rain has stopped falling, one can check the height of the water in the tube.

In our problem, the tube unfortunately has a leak at height L millimeters (mm). If the water level is above the leak then water drains from the tube at a rate of *K* millimeters per hour (mm/h).

We want to figure out how much rain fell during a particular rainfall. We assume that the tube is high enough that it does not overflow. We also assume that rain falls at an (unknown) uniform rate during a rainfall, and that water does not evaporate from the tube. The height of the leak itself is also negligible.

The first line of the input file contains an integer N (N < 40) which denotes the total number of test cases. The description of each test case is given below:

A line with five positive numbers: $L\ K\ T_1\ T_2\ H$ where

- *L* is where the leak is (mm)
- *K* is the rate at which water leaks (mm/h)
- $T_1$ is the duration of the rainfall (h)
- $T_2$ is the time between the end of the rainfall and the observation of the water level (h)
- *H* is the water level in the tube when we observe it (mm)

Each number is at least 0.01 and at most 1000.00, and each is given with two decimals.

For each test case print one line with two floating point numbers $F_1\ F_2$ where $F_1$ is the smallest rainfall in millimeters that would result in the given observation, and $F_2$ is the largest rainfall in millimeters that would result in the given observation. Values with either absolute or relative error smaller than $10^{-6}$ are acceptable.

| Sample Input | Sample Output |
|---|---|
| 2<br>80.00 0.50 2.00 1.50 80.00<br>150.00 1.00 100.00 150.00 100.00 | 80.000000 80.759403<br>100.000000 100.000000 |

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**
Школа „Състезателно програмиране”
**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## J. Key Task

Some of the university buildings are old as well. And the navigation in old buildings can sometimes be a little bit tricky, because of strange long corridors that fork and join at absolutely unexpected places.

The result is that some first-graders have often difficulties finding the right way to their classes. Therefore, the Student Union has developed a computer game to help the students to practice their orientation skills. The goal of the game is to find the way out of a labyrinth. Your task is to write a verification software that solves this game.

The labyrinth is a 2-dimensional grid of squares, each square is either free or filled with a wall. Some of the free squares may contain doors or keys. There are four different types of keys and doors: blue, yellow, red, and green. Each key can open only doors of the same color.

You can move between adjacent free squares vertically or horizontally, diagonal movement is not allowed. You may not go across walls and you cannot leave the labyrinth area. If a square contains a door, you may go there only if you have stepped on a square with an appropriate key before.

## Input

The input consists of several maps. Each map begins with a line containing two integer numbers $R$ and $C$ ($1 \leq R, C \leq 100$) specifying the map size. Then there are $R$ lines each containing $C$ characters. Each character is one of the following:

| Character | | Meaning |
|---|---|---|
| Hash mark | # | Wall |
| Dot | . | Free square |
| Asterisk | * | Your position |
| Uppercase letter | B Y R G | Blue, yellow, red, or green door |
| Lowercase letter | b y r g | Blue, yellow, red, or green key |
| Uppercase X | X | Exit |

Note that it is allowed to have

- more than one exit,
- no exit at all,
- more doors and/or keys of the same color, and
- keys without corresponding doors and vice versa.

You may assume that the marker of your position ("*") will appear exactly once in every map. There is one blank line after each map. The input is terminated by two zeros in place of the map size.

## Output

For each map, print one line containing the sentence "Escape possible in *S* steps.", where *S* is the smallest possible number of step to reach any of the exits. If no exit can be reached, output the string "The poor student is trapped!" instead. One step is defined as a movement between two adjacent cells. Grabbing a key or unlocking a door does not count as a step.

## Sample Input

```
1 10
*........X

1 3
*#X

3 20
####################
#XY.gBr.*.Rb.G.GG.y#
####################

0 0
```

## Sample Output

```
Escape possible in 9 steps.
The poor student is trapped!
Escape possible in 45 steps.
```

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

**Департамент Информатика**
Школа „Състезателно програмиране”
**СЪСТЕЗАНИЕ, 13 април 2013 г.**

## K. Savings Account

Suppose you open a savings account with a certain initial balance. You will not make any withdrawals or further deposits for a number of years. The bank will compound your balance (add the annual interest) once a year, on the anniversary of the opening of the account. Your goal is to achieve a certain target amount in your savings account. In how may years will the target amount be achieved?

## Input

The input will contain one test case per line. Each line will contain three numbers: the initial balance, the annual interest rate (as a percentage of the balance), and the target amount, separated by blank spaces. These will be positive numbers; they may or may not contain a decimal point. The target amount will be greater than the initial balance. The input is terminated by end-of-file.

## Output

For each line of input, your program will produce exactly one line of output: This line will contain one positive integer value: the number of years required to achieve the target amount.

## Sample Input

```
200.00 6.5 300
500 4 1000.00
```

## Sample Output

```
7
18
```