FLORIDA INTERNATIONAL UNIVERSITY

**2017 FIU High School Programming Competition**

# Problem Set


A Hidden Password

B Slikar

C Silueta

D Scaling Recipes

E Pizza Delivery

F Bear and Prime Numbers

G Gemstones

H The Amazing Human Cannonball

I Arithmetic

J Mine the Gradient

# A: Hidden Password

Insecure Inc. has decided to shift directions after a failed attempt at developing a new encryption standard. Their new effort is a password system used to hide a password inside another string of characters we denote as a message. However, it is important that the message has a certain property relative to the hidden password.

Let us assume that we denote the characters of the password as $c_1, c_2 \ldots c_P$ (although those characters need not be distinct). To be a valid message for the password, if you start from the beginning of the message and search for any character from the set $\{c_1, c_2 \ldots c_P\}$, $c_1$ must be the first character found. Subsequently, if you continue looking from that point in the message for any character from the set $\{c_2 \ldots c_P\}$, $c_2$ must be the next character found. Continuing in that manner, $c_3$ must be the next character found from the set $\{c_3 \ldots c_P\}$, and so on until reaching $c_P$.

For example, if the password is ABC, then the string HAPPYBIRTHDAYCACEY is a valid message.

- Notice that A is the first of the set {{A, B, C}} to appear in the message. (The initial H is not relevant.)
- Following the A that was found, the next occurrence from the set {{B, C}} is B.
- Following the B that was found, the next occurrence from the set {{C}} is indeed C.

(Note that the A in DAY is not relevant, since we are only looking for a C at this point, and the additional A and C in CACEY are not relevant, because we have already completed the password with the first C.) However, for the password ABC, the string TRAGICBIRTHDAYCACEY is not a valid message. While the A is the first of the set {{A, B, C}} to appear in the string, the next occurrence from the set {{B, C}} is C rather than B. Also, the string HAPPYBIRTHDAY is not a valid message for the password ABC because the C never appears.

As an example with duplicate letters in the password, consider the password SECRET. For this password, the string SOMECHORESARETOUGH is a valid message. In contrast, the string SOMECHEERSARETOUGH is not a valid message, because an extraneous E is found at the point when an R is first expected.

## Input

The input begins with $T$, the number of test cases, on a single line. This is followed by $T$ lines, each containing two strings. The first string is the password, having length $P$, with $3 \leq P \leq 8$. The second string has length $S$, with $10 \leq S \leq 40$. Both strings will consist of only uppercase letters. (That is, neither string can include whitespace, lowercase letters, digits, or other special characters.)

## Output

For each test case, output a single line with the word PASS if the second string is a valid message for the password, or FAIL otherwise.

Sample Input

```
5
ABC HAPPYBIRTHDAYCACEY
ABC TRAGICBIRTHDAYCACEY
ABC HAPPYBIRTHDAY
SECRET SOMECHORESARETOUGH
SECRET SOMECHEERSARETOUGH
```

Sample Output

```
PASS
FAIL
FAIL
PASS
FAIL
```

# B: Slikar

The evil emperor Cactus has in his possession the Magic Keg and has flooded the Enchanted Forest! The Painter and the three little hedgehogs now have to return to the Beaver's den where they will be safe from the water as quickly as possible!

The map of the Enchanted Forest consists of R rows and C columns. Empty fields are represented by '.' characters, flooded fields by '*' and rocks by 'X'. Additionally, the Beaver's den is represented by 'D' and the Painter and the three little hedgehogs are shown as 'S'.

Every minute the Painter and the three little hedgehogs can move to one of the four neighbouring fields (up, down, left or right). Every minute the flood expands as well so that all empty fields that have at least one common side with a flooded field become flooded as well. Neither water nor the Painter and the three little hedgehogs can pass through rocks. Naturally, the Painter and the three little hedgehogs cannot pass through flooded fields, and water cannot flood the Beaver's den.

Write a program that will, given a map of the Enchanted Forest, output the shortest time needed for the Painter and the three little hedgehogs to safely reach the Beaver's den. *Note:* The Painter and the three little hedgehogs cannot move into a field that is about to be flooded (in the same minute).

## Input

The first line of input will contain a single integer N that indicates the number of test cases to follow. Each test case will contain a line of input containing two positive integers, R and C, both <= 50. Following this line will be R lines, each containing C characters ('.', '*', 'X', 'D' or 'S'). The map will contain exactly one 'D' character and exactly one 'S' character.

## Output

Output the shortest possible time needed for the Painter and the three little hedgehogs to safely reach the Beaver's den. If this is impossible, your program should output the word "KAKTUS" on a line by itself. *Clarification of the second sample test case below:* The best they can do is to go along the lower border and then the left border, and get flooded one minute before reaching the den.

| Sample Input | Sample Output |
|---|---|
| 3<br>3 3<br>D.*<br>...<br>.S.<br>3 3<br>D.*<br>...<br>..S<br>3 6<br>D...*.<br>.X.X..<br>....S* | 3<br>KAKTUS<br>6 |

# C: Silueta

The main hero of this task, painter Vincent, spent a great deal of his youth travelling the world. Sights from numerous voyages have often been the inspiration for his, nowadays highly praised, works of art. On one occasion, Vincent found himself in a metropolis full of skyscrapers so he got down to work right away, intoxicated by the marvelous sight. For a number of reasons, incomprehensible to an average programmer, Vincent decided to paint only the silhouettes of the skyscrapers seen before him. Unfortunately, a week after he finished this masterpiece, the painting spontaneously caught fire. In order to reconstruct the painting, Vincent sought help in all directions; architects provided him with the exact dimensions of the skyscrapers, physicists ignored air resistance, mathematicians mapped everything onto a plane and now it's your turn!

From your perspective, Vincent's skyscrapers are rectangles whose sides are parallel to coordinate axes and with one side that lies on the abscissa. Part of the abscissa on the image should be shown with the characters '*', the silhouettes of the skyscrapers with '#' and fill the rest of the image with '.'. The left edge of the image must begin with a skyscraper, whereas the right edge of the image must end with a skyscraper. Additionally, in order to verify the results the mathematicians got, output the perimeter of the given silhouette not calculating the sides that lie on the abscissa. In the figure below, the dark gray line denotes the skyscraper's silhouette (character '#'), and the light gray line is the part of the abscissa located on Vincent's painting (character '*').

## Input

The first line of input contains T, the number of test cases to follow. Each test case is formatted as follows: The first line of each test case contains an integer $N$ ($1 \le N \le 100$) indicating the number of skyscrapers. Each of the next N lines contains three integers $L_i$, $R_i$ and $H_i$. Note: $1 \le L_i, R_i, H_i \le 1000$, and $3 \le (R_i - L_i) \le 1000$. These three integers describe the position of the i-th skyscraper, which in a Cartesian coordinate system, is considered a rectangle with its lower left corner in $(L_i,0)$ and upper right corner in $(R_i,H_i)$.

## Output

The first line of output must contain the perimeter of Vincent's silhouette. The next h+1 lines, where h+1 is the height of the highest skyscraper, must contain Vincent's drawing as described in the task.

Sample Input

```
2
3
1 5 4
7 11 3
9 13 5
6
2 8 7
5 13 5
2 18 3
23 26 5
20 31 7
21 30 10
```

Sample Output

```
28
........####
####....#..#
#..#..###..#
#..#..#....#
#..#..#....#
************
61
...................########.
...................#.......#.
...................#.......#.
######............##.......##
#....#............#........#
#....######.......#........#
#........#.......#........#
#........######..#........#
#..............#..#........#
#..............#..#........#
*****************************
```

# D: Scaling Recipes

A recipe is a list of ingredients and a set of instructions to prepare a dish. It is often written for a particular number of portions. If you have a recipe for 4 portions and you want to make 6 portions, it turns out that simply multiplying the amounts for each ingredient by 1.5 is often wrong! The reason is that the original recipe may have been rounded to the nearest teaspoon, gram, etc., and the rounding errors magnify when a recipe is scaled.

**Table 1**: Example Recipe

| Ingredient | Weight (g) | Percentage (%) |
|---|---|---|
| Olive Oil | 50.9 | 11.2 |
| Garlic | 12.0 | 2.7 |
| Beef | 453.6 | 100.0 |
| Onions | 1134.0 | 250.0 |
| Raisins | 82.5 | 18.2 |
| Bouillon | 10.0 | 2.2 |

Some recipes are specifically written to ease the task of scaling. These recipes are developed using "Baker's percentages." Each ingredient is listed not only by weight (in grams), but also as a percentage relative to the "main ingredient." The main ingredient will always have a 100% Baker's percentage. Note that the sum of the Baker's percentages from all ingredients is greater than 100%, and that the Baker's percentages of some ingredients may exceed 100%.

To scale a recipe, do the following steps, in order:

1. Determine the scaling factor by dividing the number of desired portions by the number of portions for which the recipe is written;

2. Multiply the weight of the main ingredient with a 100% Baker's percentage by the scaling factor. This is the scaled weight of the main ingredient;

3. Calculate the scaled weight of every other ingredient by multiplying its Baker's percentage by the scaled weight of the main ingredient.

## Input

The first line of input specifies a positive integer $T \leq 20$, consisting of the cases to follow. Each case starts with a line with three integers $R$, $P$, and $D$: $1 \leq R \leq 20$ is the number of ingredients, $1 \leq P \leq 12$ is the number of portions for which the recipe is written, and $1 \leq D \leq 1000$ is the number of desired portions. Each of the next R lines is of the form:

```
<name> <weight> <percentage>
```

where <name> is the name of the ingredient (an alphabetic string of up to 2020 characters with no embedded spaces), <weight> is the weight in grams for that ingredient, and <percentage> is its Baker's percentage. Both <weight> and <percentage> are floating-point numbers with exactly one digit after the decimal point. Each recipe will only have one ingredient with a Baker's percentage of 100%.

## Output

For each case, print the Recipe # followed by a space and the appropriate case number (see sample output below). This is followed by the list of ingredients and their scaled weights in grams. The name of the ingredient and its weight should be separated by a single space. Each ingredient is listed on its own line, in the same order as in the input. After each case, print a line of 40 dashes ('-'). Round your answers to the nearest 0.1g.

## Sample Input

```
2
6 4 20
palmOil 50.9 11.2
ginger 12.0 2.7
lamb 453.6 100.0
shallots 1134.0 250.0
figs 82.5 18.2
beefSoup 10.0 2.2
4 5 8
Milk 265.0 93.0
DisodiumCitrate 11.0 4.0
SharpCheddar 285.0 100.0
Macaroni 240.0 84.0
```

## Sample Output

```
Recipe # 1
palmOil 254.0
ginger 61.2
lamb 2268.0
shallots 5670.0
figs 412.8
beefSoup 49.9
----------------------------------------
Recipe # 2
Milk 424.1
DiodiumCitrate 18.2
SharpCheddar 456.0
Macaroni 383.0
----------------------------------------
```

# E: Pizza Delivery

Your Irish pizza and kebab restaurant is doing very well. Not only is the restaurant full almost every night, but there are also an ever increasing number of deliveries to be made, all over town. To meet this demand, you realize that it will be necessary to separate the delivery service from the restaurant. A new large kitchen, only for baking pizzas and being a base for deliveries, has to be established somewhere in town.

The main cost in the delivery service is not the making of the pizza itself, but the time it takes to deliver it. To minimize this, you need to carefully plan the location of the new kitchen. To your help you have a database of all last year's deliveries. For each location in the city, you know how many deliveries were made there last year. The kitchen location will be chosen based on the assumption that the pattern of demand will be the same in the future.

Your city has a typical suburban layout – an orthogonal grid of equal-size square blocks. All places of interest (delivery points and the kitchen) are considered to be located at street crossings. The distance between two street crossings is the Manhattan distance, i.e., the number of blocks you have to drive vertically, plus the number of blocks you have to drive horizontally. The total cost for a delivery point is its Manhattan distance from the kitchen, times the number of deliveries to the point. Note that we are only counting the distance from the kitchen to the delivery point. Even though we always drive directly back to the kitchen after a delivery is made, this (equal) distance is not included in the cost measure.

## Input

On the first line, there is a number, $1 \leq n \leq 20$, indicating the number of test cases. Each test case begins with a line containing two integers $x$ and $y$ ($1 \leq x, y \leq 1000$), indicating the width and height of the two-dimensional street grid. Next follow $y$ lines, each containing $x$ integers indicating the number of deliveries $d$ ($0 \leq d \leq 1000$) made to each street crossing last year.

## Output

For each test case, output the least possible total delivery cost (the sum of all delivery costs last year), assuming that the kitchen was located optimally. There should be one line for each test case, with an integer indicating the cost, followed by a single space and the word "blocks".

## Sample Input

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

## Sample Output

```
55 blocks
162 blocks
```

# F: Bear and Prime Numbers

Recently, the bear started studying data structures and faced the following problem. You are given a sequence of integers $x_1, x_2, ..., x_n$ of length $n$, and $m$ queries, each characterized by two integers $l_i$ and $r_i$. Let's introduce $f(p)$ to represent the number of such indexes $k$, such that $x_k$ is divisible by $p$. The answer to the query $l_i, r_i$ is the sum of all $f(p)$, where $p$ is a member of the set of prime numbers from segment $[l_i, r_i]$ (both borders are included in the segment). Help the bear cope with the problem.

## Input

The first line contains $T$, the number of test cases ($1 \le T \le 10$). Each of the next $T$ lines contains the following data: The first line contains integer $n$ ($1 \le n \le 10^6$). The second line contains $n$ integers $x_1, x_2, ..., x_n$ ($2 \le x_i \le 10^7$). The numbers are not necessarily distinct. The third line contains integer $m$ ($1 \le m \le 50{,}000$). Each of the following m lines contains a pair of space-separated integers, $l_i$ and $r_i$ ($2 \le l_i \le r_i \le 2*10^9$) — the numbers that characterize the current query.

## Output

For each test case, print the following: The word "Case ", followed by the integer case number, followed by a colon (:). On the next m lines, print $m$ integers — the answers to the queries, in the same order the queries appear in the input.

| Sample Input | Sample Output |
|---|---|
| 2<br>6<br>5 5 7 10 14 15<br>3<br>2 11<br>3 12<br>4 4<br>7<br>2 3 5 7 11 4 8<br>2<br>8 10<br>2 123 | Case 1:<br>9<br>7<br>0<br>Case 2:<br>0<br>7 |

*Notes:* Consider the first sample, which contains 3 queries.

- The first query is $l = 2$, $r = 11$. Sum = $f(2) + f(3) + f(5) + f(7) + f(11) = 2 + 1 + 4 + 2 + 0 = 9$.
- The second query is $l = 3$, $r = 12$. Sum = $f(3) + f(5) + f(7) + f(11) = 1 + 4 + 2 + 0 = 7$.
- The third query is $l = 4$, $r = 4$. This interval has no prime numbers, so the sum equals 0.

# G: Gemstones

If you've ever studied Geology in the field, it's a fascinating experience. For example, you might collect some rocks, take them back to the lab and try to figure out what you found. Each rock is composed of various elements, so to make it simple, we will assume that each possible element is represented by a lower-case Latin letter from 'a' to 'z'. Furthermore, an element can be present multiple times in the same rock. An element is called a *gem-element* if it occurs at least once in every rock (from your collection). Here is your task: given a list of rocks with their compositions, display the number of gem-elements that exist in those rocks.

## Input

The first input line contains $T$ ($1 \leq T \leq 20$), the number of test cases. The remaining lines contain test cases. Each test case will contain the following input: The first line consists of an integer, $N$ ($1 \leq N \leq 100$), the number of rocks in the sample collection. Each of the next $N$ lines contains the composition of a single rock. Each composition consists of only lower-case English letters, and its length is between 1 and 100.

## Output

For each test case, print the number of gem-elements common to all the rocks. If there are none, print 0 (zero). For example, in the first test case shown below, "a" and "b" are the only two types of gem-elements, since these are the only characters that occur in every rock's composition.

## Sample Input

```
2
3
abcdde
baccd
eeabg
4
xbcxxe
bayxdc
eyxbgc
abxccy
```

## Sample Output

```
2
3
```

© HackerRank 2017

# H: The Amazing Human Cannonball

The amazing human cannonball show is coming to town, and you are asked to double-check their calculations to make sure no one gets injured! The human cannonball is fired from a cannon that is a distance $x_1$ from a vertical wall with a hole through which the cannonball must fly. The lower edge of the hole is at height $h_1$ and the upper edge is at height $h_2$. The initial velocity of the cannonball is given as $v_0$ and you also know the angle $\theta$ of the cannon relative to the ground.

Thanks to their innovative suits, human cannonballs can fly without air resistance, and thus their trajectory can be modeled using the formulas shown here, where $x(t)$, $y(t)$ provides the position of a cannon ball at time $t$ that is fired from point (0,0). $g$ is the acceleration due to gravity ($g$ = 9.81 m/s$^2$).

$$x(t) = v_0 t \cos\theta$$
$$y(t) = v_0 t \sin\theta - \frac{1}{2} g t^2$$

Write a program to determine if the human cannonball can make it safely through the hole in the wall. To pass safely, there has to be a vertical safety margin of 1 m both below and above the point where the ball's trajectory crosses the centerline of the wall.

## Input

The input will consist of up to 100 test cases. The first line contains an integer $N$, denoting the number of test cases that follow. Each test case has 5 parameters: $v_0$, $\theta$, $x_1$, $h_1$, $h_2$, separated by spaces. $v_0$ ($0 < v_0 \leq 200$) represents the ball's initial velocity in m/s. $\theta$ is an angle given in degrees ($0 < \theta < 90$), $x_1$ ($0 < x_1 < 1000$) is the distance from the cannon to the wall, $h_1$ and $h_2$ ($0 < h_1 < h_2 < 1000$) are the heights of the lower and upper edges of the wall, respectively. All numbers are floating point numbers.

## Output

Each line of output is a separate test case. If the cannon ball can safely make it through the wall, output "**`Safe`**". Otherwise, output "**`Not Safe`**".

| Sample Input | Sample Output |
|---|---|
| 11<br>19 45 20 9 12<br>20 45 20 9 12<br>25 45 20 9 12<br>20 43 20 9 12<br>20 47.5 20 9 12<br>20 45 17 9 12<br>20 45 24 9 12<br>20 45 20 10 12<br>20 45 20 9 11<br>20 45 20 9.0 11.5<br>20 45 18.1 9 12 | Not Safe<br>Safe<br>Not Safe<br>Not Safe<br>Not Safe<br>Not Safe<br>Not Safe<br>Not Safe<br>Not Safe<br>Safe<br>Safe |

© open.kattis.com

# I: Arithmetic

Ever wonder what will happen when you meet a person who uses a different base system to talk about numbers? For example, when you hear someone saying this mysterious number 1010110001 (which is in binary), can you quickly convert it to its decimal form, 689?

In Harkerland, everyone uses the octal numeral system (base 8) when talking about numbers. Therefore, new immigrants often have difficulties parsing numbers quickly. One of them is Alex, who used to live in a place where the hexadecimal numeral system (base 16) was adopted. In order to adapt to the Harkerland lifestyle as soon as possible, he wants to perform instant octal-to-hexadecimal conversion. Naturally, Alex wants you to use a computer program to accomplish the base conversion.

## Input

The first line of input indicates the number of test cases to follow. Then for each test case, input consists of a single non-negative integer k (k > 0), written in base 8. The integer will have no leading zeros and will be less than $8^{2000}$.

## Output

For each test case, output the same integer, written in base 16. The output must not include any extra leading zeroes. Use capital letters A, B, ……, F to represent the values 10,11,…,15 in hexadecimal. For the sample input test case #1 below, $4444_8$ equals $2340_{10}$, which also equals $924_{16}$.

## Sample Input

```
4
4444
20
3211
7654321001234566
```

## Sample Output

```
924
10
689
FAC688053976
```

# J: Mine the Gradient

If we need to be prepared for the possible danger of an alien invasion, the first thing to do is to find out where can they come from. One way to determine if a planet is inhabited by intelligent creatures is to study high resolution pictures of planets trying to find typical characteristics of effects of intelligent life. There are so many habitable planets that a computer program must be written for this task. One distinctive feature of a colonized planet is the presence of surface mines. An alien mine is a square structure with a depth uniformly decreasing from one of the sides of the square towards the opposite.

Your task is to find the largest mine on a provided bitmap image of a planet. The picture is a rectangular grid of numbers between 0 and 65,535, representing the shades of grey. Mines will appear as squares of either the same shade or with shade levels gradually and uniformly changing from one side to another. See the figure for an example. Your program will consider only square mines in some special orientations.

An axis-parallel square is the set of pixels (i, j) such that $c_1 \le i \le c_2$ and $r_1 \le j \le r_2$ for some $c_1$, $c_2$, $r_1$, $r_2$, and $(c_2 - c_1) = (r_2 - r_1)$. A vertically (horizontally) oriented mine is an axis-parallel square for which there exist integers $S$ and $K$ such that the shade of every pixel $(i, j)$ of the square is equal to $S + iK$ ($S + jK$ for horizontally oriented mines). A diagonally oriented mine is an axis-parallel square for which there exist integers $Q \in \{1, -1\}$, $S$ and $K$ such that the shade of every pixel (i, j) of the square is equal to $S + (i + Qj)\,K$.

## Input

The input contains at most 20 descriptions of pictures. The first line of each description contains two numbers $N$ and $M$ ($1 \le N, M \le 2{,}000$), the height and width of the picture. The following N lines contain $M$ space-separated integers each. The $j$-th number in the $i$-th row $A_{i,j}$ ($0 \le A_{i,j} \le 65{,}535$) describes the shade of gray of the pixel in the $i$-th row and $j$-th column of the picture bitmap. The last description is followed by a line containing two zeros.

## Output

For each picture, output the area (number of pixels inside) of the largest horizontal, vertical, or diagonal mine.

| Sample Input | Sample Output |
|---|---|
| 4 4<br>10 1 13 20<br>18 9 11 13<br>5 7 9 6<br>6 5 7 7<br>3 3<br>10 1 13<br>18 9 11<br>5 1000 9<br>4 4<br>10 12 15 20<br>5 9 13 10<br>5 9 13 6<br>5 9 13 7<br>0 0 | 4<br>1<br>9 |