# Problem A
# Agents

The national intelligent agency has selected their best $n$ agents to conduct a highly classified mission. To reduce the exposure risk, the agents are organized into smaller teams working as separated units. That is members of the same team know each others but not the agents in the other teams. Once an agent is exposed, all his/her team members will get exposed as well. Given the sizes of the teams, the risk is measured as the expected number of exposed agents, once an arbitrary agent (uniformly selected among $n$ agents) gets exposed. Ideally, the risk is minimum at one when $n$ agents are divided into $n$ separated teams of size one. However, dividing the agents into smaller teams also incurs operational costs computed as follows: assigning any agent to the $i$th team incurs a cost $c_i \geq 0$, for $1 \leq i \leq n$ (obviously, the maximum number of teams is $n$). Given the number of agents $n$, a risk level $\mu$(theexpected number of exposed agents) that the agency can tolerate, and the costs $c_1, c_2, \dots, c_n$, find a way to assign agents into teams to minimize the operational costs while ensuring that the risk does not exceed $\mu$.

**Input**
Each line contains a test case, starting with an integer $n$ ($1 \leq n \leq 500$) indicating the number of agents. Following this are $n$ non-negative real numbers $c_1, c_2, \dots, c_n$ in which $c_i$ specifies the cost of assigning an agent to the $i$-th team ($0 \leq c_i \leq 100$). The line ends with a real number $1 \leq \mu \leq n$ indicating the maximum risk level that the agency can tolerate. The input terminates with a zero and you do not have to process this case.

**Output**
For each test case, display a single line containing the case number and the minimum operational cost (formatted like the sample output). All the operational costs should be accurate to three decimal places.

| Sample input | Output for sample input |
| --- | --- |
| `4 2.0 3.7 1.2 1.8 2`<br>`6 6.2 7.9 2.4 3.2 1.2 3.2 1`<br>`5 1.2 3.5 9.1 7.3 9.6 5`<br>`0` | `Case 1: 6.000`<br>`Case 2: 24.100`<br>`Case 3: 6.000` |

# Problem B
# Bob and Alice

Alice and Bob are playing a simple game. They first start with a positive integer N, and then they take alternative turns. In each turn, they chose a positive integer from 1 to K, and subtract it from N. After ones turn, if N equals to 0, he/she is the winner. Alice will take the first turn. Bob and Alice are very intelligent and will always play with optimal strategy.

For example if N = 5, K = 3. At the first turn, Alice is able to choose 1, 2 or 3:

- If Alice chooses 1 at her first turn and makes N reduced to 4. Now Bob can chose 1, 2 or 3 to reduce N to 3, 2 or 1. Whatever number he chooses, Alice will be able to reduce N to zero in her next turn.
- If Alice chooses 2 or 3 at her first turn, N will be reduced to 3 or 2. Now Bob can reduce N to zero in a single turn and become a winner.

Because Alice is very intelligent, she obviously knows she should choose 1 at her first turn. Bob is also very intelligent but in that situation, he doesn't have a chance. Given N and K, your task is to determine who the winner of this game is, if both Alice and Bob play with optimal strategy.

**Input**
The input consists of several test cases. Each test case is printed in one line with the number N and K. The input will terminate with N=0 and K=0 and you don't have to process this case. All the numbers in the input are not exceed one million.

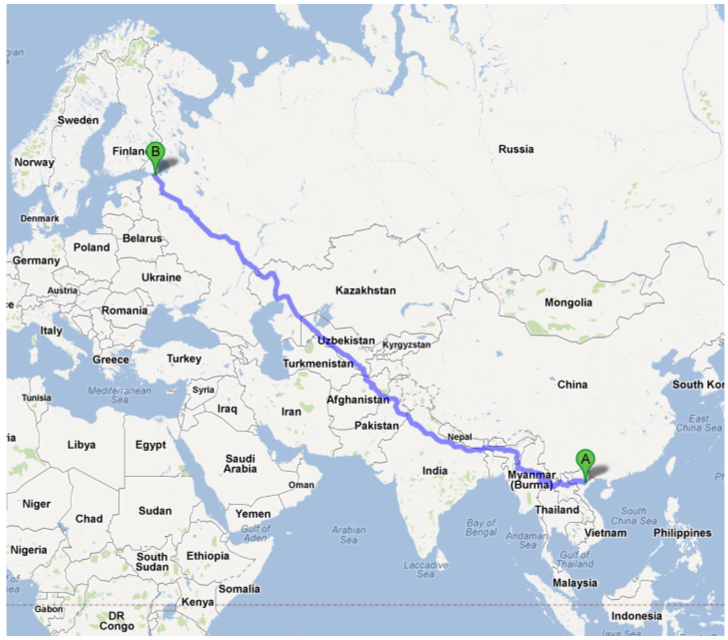
**Output**
Each answer of a test case should be printed in a single line. The answer should be either:

- Alice will win
- Bob will win
- Can not determine

| Sample input | Output for sample input |
|---|---|
| `5 3`<br>`4 3`<br>`0 0` | `Alice will win`<br>`Bob will win` |

# Problem C
# Crazy road trip

Despite the ACM National Contest is still underway, team number X is thinking of a road trip from Hanoi to St Petersburg to take part in the ACM/ICPC World Final 2013. It's going to be a long, fantastic adventure.




They plan to drive at most L kilometers each day; stay and rest at night in a city. Even these students are a little crazy; they don't think sleeping by the road is a good idea. There are N cities which they can find a decent hotel; these cities are connected by M two-way roads. These cities are numbered from 1 to N; Hanoi is numbered 1 and St Petersburg is numbered N.

Your task is to calculate in at least how many days they can reach St Petersburg.

### Input

The first line of input is the number of test T. Then T tests follow. Each test starts with a line consists of the number of cities – N, the number of roads – M and the maximum driving distance for a day – L ($2 \le N \le 10000$, $0 \le M \le 100000$, $0 \le L \le 1000$). Each of the next M lines consists of three integers U V W – which means there is a two-way road between U and V with the distance of W kilometers ($1 \le W \le 1000$).

### Output

For each test, print one number indicating the minimal day of the trip or print NO SOLUTION if it is impossible to plan a satisfied trip.

| Sample input | Output for sample input |
|---|---|
| 2<br>4 3 50<br>1 2 30<br>2 3 20<br>3 4 30<br>4 3 25<br>1 2 30<br>2 3 20<br>3 4 30 | 2<br>NO SOLUTION |

# Problem D
# Data Analysis

Over the last few months, several deadly holes had appeared in many roads all over a country. Experts explained that one possible cause of these events is the lack of quality in roads and highways. Thus, the government decided to do an inspection on the most important highway of the country. They took samples of N points along the highway, evaluate and give marks on quality of these samples. Now, gathering all the data, we have to analyze all of these data. The first step in analyze these data is to find all the abnormal points. We define a point abnormal if it is smaller or greater than both of its 2 neighbors: previous point and next point. If a point is smaller/larger than its neighbors, we call it local minimum/maximum. Since point #1 and point #N only have one neighbor, we do not analyze these two points. Your task is to analyze how many local minimum and local maximum there are.

**Input**

The input consists of several test cases. Each test cases starts with a number N ($2 \le N \le 1000$) indicate the number of sample points, followed by N positive integers indicating the marks of these points. The input terminates with a number 0 and you don't have to process this case. All the number in the input does not exceed one million.

**Output**

For each test in the input, print the number of local minimum points and the number of local maximum.

| Sample input | Output for sample input |
|---|---|
| `4 1 2 1 3`<br>`3 3 3 3`<br>`0` | `1 1`<br>`0 0` |

# Problem E
# Equal probability

In this problem, we are given a robot and a maze. A maze comprises of three types of cells:

- Empty cells: "."
- Block cells: "#"
- Return cells: "R"

One of the empty cells is the starting cell "S" of the robot; another is its target cell "T". The robot is allowed to move up, down, left, right; but not to move outside the maze, not to step into a block cell "#", or not to stay in the current cell. If the robot happens to step into a return cell "R", by some magic, it will immediately be teleported back to the starting cell "S".

The objective of the robot is to arrive at the target cell "T". Since the structure of the maze is not provided to the robot, at every point, it chooses at equal probability for every valid move allowed. Note that the robot may even step back into its starting cell - it simply chooses each move at random.

Your task is to find the average number of moves the robot will take to reach the target cell. It is guaranteed that there exists a path from the starting cell "S" to the target cell "T".

**Input**
The input has many test cases. Each test case starts with the number of rows m and the number of columns n of the maze. The next *m* lines describe the maze. The last test case is followed by two number zeros.

**Output**
For each test case, output the average number of moves the robot will take. Output the result with precisely 7 digits after the decimal points.

**Limits**
The size of each maze is not bigger than 16 by 16.
There will not be more than 100 test cases.

| Sample input | Output for sample input |
|---|---|
| 3 3<br>##.<br>#ST<br>##.<br>3 3<br>#R.<br>RST<br>#R.<br>3 3<br>#R.<br>.ST<br>#R.<br>3 4<br>...#<br>#ST.<br>....<br>0 0 | 1.0000000<br>4.0000000<br>5.0000000<br>6.0357143 |

**Comments on sample test cases**

In the first case, there's only one option to move, and the robot is magically at its target!

In the second case, only one fourth of the chance can reach the target cell. This means the robot wastes three third of its efforts by jumping into "R" and being teleported back to the starting cell. Therefore, its average number of moves is 4.

In the third case, if the robot chooses to move left, it will waste more since the next move is to simply go back to the starting cell. The result of the third case must be greater than that of the second case. Note that the two empty cells in the upper and lower right can never be reached.

In the fourth case, even though the starting and target cells are next to each other, the robot may wander around before finally settling down. That's the cost of not providing the maze to the robot!

# Problem F
# Finding food

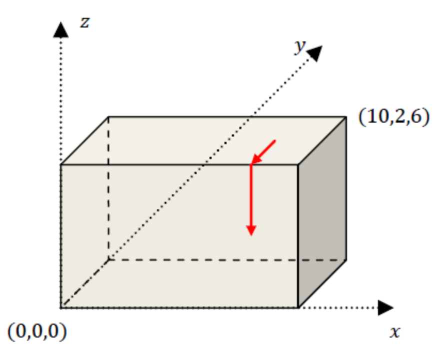
Carl – the ant is a special ant. It has a very good sense of space and direction, so it can always travel from point to point by the shortest possible way. These days, Carl spends time around a candy box which is in shape of a cuboid. He is wandering around 5 faces of the box trying to find a hole where he can enter the box. This candy box is put on a table so its bottom face is inaccessible. When Carl got some information from other ants about a point where he can enter the candy box, he immediately goes to that point via the shortest possible way.

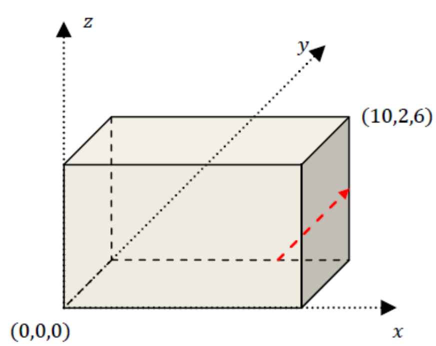
If we will describe the candy box in geometry languages using the Cartesian coordinate system , the candy box is a cuboid with all faces parallel to planes $x = 0$ or $y = 0$ or $z = 0$. The two opposite corner are at (0,0,0) and $(x_0, y_0, z_0)$. The current position of Carl is $(x_1, y_1, z_1)$ and he has just got information about a hole at $(x_2, y_2, z_2)$. Your task is to calculate the distance Carl will travel.

**Input**
The input contains several test cases. Each test case consists of 9 positive integers $x_0$, $y_0$, $z_0$, $x_1$, $y_1$, $z_1$, $x_2$, $y_2$, $z_2$. The input terminates with 9 zeros. It is guarantee that these integer are less than or equal to 100; point $(x_1, y_1, z_1)$ and $(x_2, y_2, z_2)$ are on accessible surfaces of the box.

**Output**
For each test in the input, print the distance Carl will travel. Print the result with exactly six digits to the right of the decimal point.

| Sample input | Output for sample input |
|---|---|
| 10 2 6 8 1 6 8 0 3<br>10 2 6 7 2 0 10 2 3<br>0 0 0 0 0 0 0 0 0 | 4.000000<br>4.242641 |



# Problem G
# General string

The general string of a non-empty set S which contains N non-empty strings is the string that takes all elements of S as a substring. For example, a set {“ACM”, “ICPC”, ”CONTEST”} has many general strings such as:

- ACMICPCCONTEST
- ACMICPCCONTESTPROGRAMMING
- ACMICPCONTEST
- CONTESTACMICPC

but not “CONACMICPCTEST” because this string does not take “CONTEST” as a substring.

Given a set S, your task is to calculate a shortest general string of S. These strings contain only uppercase characters. If there is more than one answer, return the one that come first lexicographically.

**Input**
The first line of the input is the number of test T (T ≤ 30). Then T tests follow. Each test is printed on one line, containing all the string of the set. These strings are separated by space(s) or tab(s). There are at most 8 strings and each string is not longer than 10 characters.

**Output**
For each test in the input, you should print a general string of that test.

| **Sample input** | **Output for sample input** |
|---|---|
| `2`<br>`              AB     BC     CA`<br>`ACM   ICPC   CONTEST` | `ABCA`<br>`ACMICPCONTEST` |

# Problem H
# How to transpose a matrix

In a computer memory, each matrix $A$ with size $M \times N$ is represented by a 2-dimension array $a$, where $a[i][j]$ represent an element in row $i$, column $j$ (rows are numbered $1, 2, \ldots, M$, columns are numbered $1,2, \ldots, N$).

A transposition of matrix $A$ to matrix $B$ is defined: if $A$ has size of $M \times N$ then $B$ has size of $N \times M$ and $a[i][j] = b[j][i]\ \forall i = 1 \div M, \forall j = 1 \div N$.

In a computer memory, multi-dimension array is stored like a linear array. For example, matrix $A = \begin{bmatrix} 1\ 2\ 3\ 4 \\ 5\ 6\ 7\ 8 \end{bmatrix}$ is stored as array $S_A = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$. Transposition matrix of $A$ is $B = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{bmatrix}$ will be stored as array $S_B = [1\ 5\ 2\ 6\ 3\ 7\ 4\ 8]$.

In terms of computer memory, transposing matrix $A$ means to rearrange elements of $S_A$ to get $S_B$ by a series of pair-swap operations. Given $M, N$ as the size of $A$, you are to calculate the minimum number of pair-swap operations in order to transpose $A$.

### Input

The input contains several tests, each is written in a line consists of two integers $M, N$ ($1 \leq M, N \leq 5000$). The input terminates with two zeros and you don't have to process this case.

### Output

For each test in the input, print the minimal number of pair-swap operations.

| Sample input | Output for sample input |
|---|---|
| 2 2<br>2 4<br>0 0 | 1<br>4 |