Problem A

Add All

Given a set of numbers your goal is to add them all, while minimizing the cost of the addition. At any point, you may choose to add any two of the integers. The cost of doing so is simply the sum of the two numbers. For example, the cost of adding 1 and 10 is 11. 3, 6 and 2 can be added in several different ways. The minimum cost comes from adding 2+3 first to yield 5 and 5+6 to yield 11. The total cost for these two operations is 16 (5 + 11). Given a sequence of numbers calculate the minimum cost to add them into a single integer.

Input

First line of the input contains T the number of test cases. First line of each test case contains N the number of integers in the sequence. Second line contains N integers separated by a single space. N is between 1 and 20000 inclusive. Each of the integers in the sequence will be between 1 and 10000.

Output

For each test case output contains a single integer denoting the minimum cost.

Sample Input	Sample output
2 3 1 2 3 4 1 2 3 4	9 19

A: Balloons

As you may know, balloons are handed out during ACM contests to teams as they solve problems. However, this sometimes presents logistical challenges. In particular, one contest hosting site maintains two rooms, A and B, each containing a supply of balloons. There are **N** teams attending the contest at that site, each sitting at a different location. Some are closer to room A, others are closer to room B, and others are equally distant. Given the number of balloons needed by each team and the distance from each team to room A, and to room B, what is the minimum total possible distance that must be traveled by all balloons as they are delivered to their respective teams, assuming they are allocated in an optimal fashion from rooms A and B? For the purposes of this problem, assume that all of the balloons are identical.

The Input

There will be several test cases in the input. Each test case will begin with a line with three integers:

```
N A B
```

Where **N** is the number of teams (1 ≤ **N** ≤ 1,000), and **A** and **B** are the number of balloons in rooms A and B, respectively (0 ≤ **A**,**B** ≤ 10,000). On each of the next N lines there will be three integers, representing information for each team:

```
K DA DB
```

Where **K** is the total number of balloons that this team will need, **DA** is the distance of this team from room A, and **DB** is this team's distance from room B (0 ≤ **DA**,**DB** ≤ 1,000). You may assume that there are enough balloons – that is, Σ(**K**'s) ≤ **A**+**B**. The input will end with a line with three 0s.

The Output

For each test case, output a single integer, representing the minimum total distance that must be traveled to deliver all of the balloons. Count only the outbound trip, from room A or room B to the team. Don't count the distance that a runner must travel to return to room A or room B. Print each integer on its own line with no spaces. Do not print any blank lines between answers.

Sample Input

```
3 15 35
10 20 10
10 10 30
10 40 10
0 0 0
```

Sample Output

```
300
```

Problem B

Containers

Input File: containers.in

A seaport container terminal stores large containers that are eventually loaded on seagoing ships for transport abroad. Containers coming to the terminal by road and rail are stacked at the terminal as they arrive.

Seagoing ships carry large numbers of containers. The time to load a ship depends in part on the locations of its containers. The loading time increases when the containers are not on the top of the stacks, but can be fetched only after removing other containers that are on top of them.

The container terminal needs a plan for stacking containers in order to decrease loading time. The plan must allow each ship to be loaded by accessing only topmost containers on the stacks, and minimizing the total number of stacks needed.

For this problem, we know the order in which ships must be loaded and the order in which containers arrive. Each ship is represented by a capital letter between A and Z (inclusive), and the ships will be loaded in alphabetical order. Each container is labeled with a capital letter representing the ship onto which it needs to be loaded. There is no limit on the number of containers that can be placed in a single stack.

Input

The input file contains multiple test cases. Each test case consists of a single line containing from 1 to 1000 capital letters representing the order of arrival of a set of containers. For example, the line ABAC means consecutive containers arrive to be loaded onto ships A, B, A, and C, respectively. When all containers have arrived, the ships are loaded in strictly increasing order: first ship A, then ship B, and so on.

A line containing the word `end` follows the last test case.

Output

For each input case, print the case number (beginning with 1) and the minimum number of stacks needed to store the containers before loading starts. Your output format should be similar to the one shown here.

Sample Input	Output for the Sample Input
A	Case 1: 1
CBACBACBACBACBA	Case 2: 3
CCCCBBBBAAAA	Case 3: 1
ACMICPC	Case 4: 4
end	

Minimum Diaper Cost

Diapers are expensive! Arup has noticed that if he always changes his daughter's diapers right when they get a bit wet, he might be wasting money because often times, a few minutes later, Arup has to change the diaper he just changed! Thus, it might be more effective to change a diaper only when it's been filled closer to capacity, so to speak.[1] In this program, you'll help Arup determine the minimum amount of money he must spend on diapers, given a list of the size of his daughter's waste, in the order produced. In order to solve this problem, you must know the capacity of each diaper in milliliters, as well as the cost of diapers. Diapers may only be bought in boxes and not individually. Thus, if a box has 108 diapers and costs $32.99 and you only need 135 diapers, your cost would be $65.98, the cost of two boxes, since you must buy an integer number of boxes.

In order to simplify the problem, we'll assume that we want the minimum cost given that only one type of box of diapers will be bought, since it's too confusing to buy some boxes of one type and other boxes of another type.

Input

The first line of input will contain a single positive integer, T $(T \leq 50)$, the number of input cases. The first line of input of each test case will contain a positive integer, N $(1 < N < 10000)$, representing the number of different items of waste, followed by a positive integer D $(D \leq 10)$, representing the number of different types of boxes of diapers. The next D lines of each test case will contain information about one box of diapers each. Each of these lines will contain three values separated by spaces: the number of diapers in that type of box (a positive integer ≤ 300), the price for that type of box in dollars (a positive real number to two decimal places ≤ 100) and the capacity of the diaper in milliliters (a positive integer ≤ 500). The last line of each test case will contain N positive integer representing the sizes of waste produced, in milliliters. Note: Each of the values on this last line will be less than or equal to the capacities of at least one of the boxes of diapers available for the case, ensuring that at least one type of diaper can handle the waste produced.

Output

For each test case, start the output a single line with the following format:

```
Diaper Scenario #k: Buying box B, you spend $X.
```

where k is the number of the test case, starting with 1, B is the box number (in between 1 and D, inclusive), and X is the cost of diapers buying box number B to handle the given diaper situation. X should be printed to two decimal places.

[1] In reality, Arup does NOT condone leaving dirty diapers on children to minimize the cost of diapers!!!

Sample Input

```
2
10 2
3 3.99 100
5 5.99 90
50 50 50 50 50 50 50 50 50 50
10 2
10 1.99 40
2 9.99 200
5 5 5 5 5 5 5 5 10 200
```

Sample Output

```
Diaper Scenario #1: Buying box 1, you spend $7.98.
Diaper Scenario #2: Buying box 2, you spend $9.99.
```

Note: In the first example, you would need 5 diapers from box 1, which means you would have to buy two of these boxes, OR you would need 10 diapers from box 2, which means you would have to buy two of these boxes. In the second example, the capacity of the first diaper can not handle the last item of waste, so by default, the only possible option is box 2.

UCF Practice Problem: Minimalist Violinist (Version 1)

Filename: violin1

Violinists must put their fingers down strategically when they play. Each note may require a different finger to be down. We typically refer to the fingers by numbers 1, 2, 3 and 4. Any subset of these four fingers may be down at any given moment of time, but the one that gets played is the maximum of the fingers that are down. For example, if the violinist had the fingers 1, 2 and 4 down, then the note played at that time would correspond to the 4. (A violinist can also be playing on any one of four strings and sometimes play a combination of those strings, but for this problem, we'll just deal with a violinist playing on a single string.) If no fingers are down, then the note played is 0.

A violinist is given a sequence of notes to play, which we can simply designated for the purposes of this problem as a sequence of numbers, each of which is 1, 2, 3 or 4. Here is an example:

0, 0, 4, 4, 1, 3, 1, 2, 1, 4, 0, 0, and 0

One way in which the violinst could play this sequence would be to put no fingers down for the first 2 notes, then place all four fingers down in preparation for the third note, left fingers 2, 3 and 4 for the fifth note, put just the third finger down for the sixth note, lift that third finger for the seventh note, put the second finger down for the eighth note, lift the second finger for the ninth note, put the fourth finger down, following by removing both the first and fourth finger for the last three notes. This particular method of playing the notes results in 4 + 3 + 1 + 1 + 1 + 1 + 1 + 2 = 14 total finger movements. Note: This isn't the minimum possible, however.

<u>The Problem</u>

Given a sequence of notes (0, 1, 2, 3 or 4) that a violinist must play, determine the minimum number of finger movements necessary to play the sequence. Assume the violinist starts with no fingers down and can end with any subset of her fingers down.

<u>The Input</u>

The first line of the input contains a single positive integer, n ($n \le 30$), representing the number of violin sequences described in the input. The cases follow, one per line. The first value on each of these lines is a positive integer, k ($k \le 100000$), representing the length of the sequence. This is followed by k space-separated integers, representing the sequence of notes to be played. Each of these values will be 0, 1, 2, 3 or 4.

<u>The Output</u>

For each sequence to be played, output the minimum total number of finger movements necessary to play the sequence on a line by itself.

<u>Sample Input</u>

```
2
13 0 0 4 4 1 3 1 2 1 4 0 0 0
3 1 4 1
```

<u>Sample Output</u>

```
10
3
```