动态规划算法

**1. Max Sum**

Problem Description

Given a sequence a[1],a[2],a[3]......a[n], your job is to calculate the max sum of a sub-sequence. For example, given (6,-1,5,4,-7), the max sum in this sequence is 6 + (-1) + 5 + 4 = 14.

Input

The first line of the input contains an integer T(1<=T<=20) which means the number of test cases. Then T lines follow, each line starts with a number N(1<=N<=100000), then N integers followed(all the integers are between -1000 and 1000).

Output

For each test case, you should output two lines. The first line is "Case #:", # means the number of the test case. The second line contains three integers, the Max Sum in the sequence, the start position of the sub-sequence, the end position of the sub-sequence. If there are more than one result, output the first one. Output a blank line between two cases.

Sample Input

2

5 6 -1 5 4 -7

7 0 6 -1 1 -6 7 -5

Sample Output

Case 1:

14 1 4

Case 2:

7 1 6

**2. Robberies**

Problem Description

The aspiring Roy the Robber has seen a lot of American movies, and knows that the bad guys usually gets caught in the end, often because they become too greedy. He has decided to work in the lucrative business of bank robbery only for a short while, before retiring to a comfortable job at a university.



For a few months now, Roy has been assessing the security of various banks and the amount of cash they hold. He wants to make a calculated risk, and grab as much money as possible.  
  
His mother, Ola, has decided upon a tolerable probability of getting caught. She feels that he is safe enough if the banks he robs together give a probability less than this.

Input

The first line of input gives T, the number of cases. For each scenario, the first line of input gives a floating point number P, the probability Roy needs to be below, and an integer N, the number of banks he has plans for. Then follow N lines, where line j gives an integer Mj and a floating point number Pj .  
Bank j contains Mj millions, and the probability of getting caught from robbing it is Pj .

Output

For each test case, output a line with the maximum number of millions he can expect to get while the probability of getting caught is less than the limit set.  
  
Notes and Constraints  
0 < T <= 100  
0.0 <= P <= 1.0  
0 < N <= 100  
0 < Mj <= 100  
0.0 <= Pj <= 1.0  
A bank goes bankrupt if it is robbed, and you may assume that all probabilities are independent as the police have very low funds.

Sample Input

3

0.04 3

1 0.02

2 0.03

3 0.05

0.06 3

2 0.03

2 0.03

3 0.05

0.10 3

1 0.03

2 0.02

3 0.05

Sample Output

2

4

6

**3.Employment Planning**

Problem Description

A project manager wants to determine the number of the workers needed in every month. He does know the minimal number of the workers needed in each month. When he hires or fires a worker, there will be some extra cost. Once a worker is hired, he will get the salary even if he is not working. The manager knows the costs of hiring a worker, firing a worker, and the salary of a worker. Then the manager will confront such a problem: how many workers he will hire or fire each month in order to keep the lowest total cost of the project.

Input

The input may contain several data sets. Each data set contains three lines. First line contains the months of the project planed to use which is no more than 12. The second line contains the cost of hiring a worker, the amount of the salary, the cost of firing a worker. The third line contains several numbers, which represent the minimal number of the workers needed each month. The input is terminated by line containing a single '0'.

Output

The output contains one line. The minimal total cost of the project.

Sample Input

3

4 5 6

10 9 11

0

Sample Output

199

**4.Attack on Titans**

Problem Description

Over centuries ago, mankind faced a new enemy, the Titans. The difference of power between mankind and their newfound enemy was overwhelming. Soon, mankind was driven to the brink of extinction. Luckily, the surviving humans managed to build three walls: Wall Maria, Wall Rose and Wall Sina. Owing to the protection of the walls, they lived in peace for more than one hundred years.

But not for long, a colossal Titan appeared out of nowhere. Instantly, the walls were shattered, along with the illusory peace of everyday life. Wall Maria was abandoned and human activity was pushed back to Wall Rose. Then mankind began to realize, hiding behind the walls equaled to death and they should manage an attack on the Titans.

So, Captain Levi, the strongest ever human being, was ordered to set up a special operation squad of N people, numbered from 1 to N. Each number should be assigned to a soldier. There are three corps that the soldiers come from: the Garrison, the Recon Corp and the Military Police. While members of the Garrison are stationed at the walls and defend the cities, the Recon Corps put their lives on the line and fight the Titans in their own territory. And Military Police serve the King by controlling the crowds and protecting order. In order to make the team more powerful, Levi will take advantage of the differences between the corps and some conditions must be met.

The Garrisons are good at team work, so Levi wants there to be at least M Garrison members assigned with continuous numbers. On the other hand, members of the Recon Corp are all elite forces of mankind. There should be no more than K Recon Corp members assigned with continuous numbers, which is redundant. Assume there is unlimited amount of members in each corp, Levi wants to know how many ways there are to arrange the special operation squad.

Input

There are multiple test cases. For each case, there is a line containing 3 integers N (0 < N < 1000000), M (0 < M < 10000) and K (0 < K < 10000), separated by spaces.

Output

One line for each case, you should output the number of ways mod 1000000007.

Sample Input

3 2 2

Sample Output

5

**5. The King’s Ups and Downs**

**Problem Description**

The king has guards of all different heights. Rather than line them up in increasing or decreasing height order, he wants to line them up so each guard is either shorter than the guards next to him or taller than the guards next to him (so the heights go up and down along the line). For example, seven guards of heights 160, 162, 164, 166, 168, 170 and 172 cm. could be arranged as:



or perhaps:



The king wants to know how many guards he needs so he can have a different up and down order at each changing of the guard for rest of his reign. To be able to do this, he needs to know for a given number of guards, n, how many different up and down orders there are:  
  
For example, if there are four guards: 1, 2, 3,4 can be arrange as:  
  
1324, 2143, 3142, 2314, 3412, 4231, 4132, 2413, 3241, 1423  
  
For this problem, you will write a program that takes as input a positive integer n, the number of guards and returns the number of up and down orders for n guards of differing heights.

**Input**

The first line of input contains a single integer P, (1 <= P <= 1000), which is the number of data sets that follow. Each data set consists of single line of input containing two integers. The first integer, D is the data set number. The second integer, n (1 <= n <= 20), is the number of guards of differing heights.

**Output**

For each data set there is one line of output. It contains the data set number (D) followed by a single space, followed by the number of up and down orders for the n guards.

**Sample Input**

4

1 1

2 3

3 4

4 20

**Sample Output**

1 1

2 4

3 10

4 740742376475050

**6.Number String**

**Problem Description**

The signature of a permutation is a string that is computed as follows: for each pair of consecutive elements of the permutation, write down the letter 'I' (increasing) if the second element is greater than the first one, otherwise write down the letter 'D' (decreasing). For example, the signature of the permutation {3,1,2,7,4,6,5} is "DIIDID".  
  
Your task is as follows: You are given a string describing the signature of many possible permutations, find out how many permutations satisfy this signature.  
  
Note: For any positive integer n, a permutation of n elements is a sequence of length n that contains each of the integers 1 through n exactly once.

**Input**

Each test case consists of a string of 1 to 1000 characters long, containing only the letters 'I', 'D' or '?', representing a permutation signature.  
Each test case occupies exactly one single line, without leading or trailing spaces.  
Proceed to the end of file. The '?' in these strings can be either 'I' or 'D'.

**Output**

For each test case, print the number of permutations satisfying the signature on a single line. In case the result is too large, print the remainder modulo 1000000007.

**Sample Input**

II

ID

DI

DD

?D

??

**Sample Output**

1

2

2

1

3

6

**7. Advanced Fruits**

**Problem Description**

The company "21st Century Fruits" has specialized in creating new sorts of fruits by transferring genes from one fruit into the genome of another one. Most times this method doesn't work, but sometimes, in very rare cases, a new fruit emerges that tastes like a mixture between both of them.  
A big topic of discussion inside the company is "How should the new creations be called?" A mixture between an apple and a pear could be called an apple-pear, of course, but this doesn't sound very interesting. The boss finally decides to use the shortest string that contains both names of the original fruits as sub-strings as the new name. For instance, "applear" contains "apple" and "pear" (APPLEar and apPlEAR), and there is no shorter string that has the same property.  
  
A combination of a cranberry and a boysenberry would therefore be called a "boysecranberry" or a "craboysenberry", for example.  
  
Your job is to write a program that computes such a shortest name for a combination of two given fruits. Your algorithm should be efficient, otherwise it is unlikely that it will execute in the alloted time for long fruit names.

**Input**

Each line of the input contains two strings that represent the names of the fruits that should be combined. All names have a maximum length of 100 and only consist of alphabetic characters.  
  
Input is terminated by end of file.

**Output**

For each test case, output the shortest name of the resulting fruit on one line. If more than one shortest name is possible, any one is acceptable.

**Sample Input**

apple peach

ananas banana

pear peach

**Sample Output**

appleach

bananas

pearch

**8. 龟兔赛跑**

Problem Description

据说在很久很久以前，可怜的兔子经历了人生中最大的打击——赛跑输给乌龟后，心中郁闷，发誓要报仇雪恨，于是躲进了杭州下沙某农业园卧薪尝胆潜心修炼，终于练成了绝技，能够毫不休息得以恒定的速度(VR m/s)一直跑。兔子一直想找机会好好得教训一下乌龟，以雪前耻。

比赛是设在一条笔直的道路上，长度为L米，规则很简单，谁先到达终点谁就算获胜。  
无奈乌龟自从上次获胜以后，成了名龟，被一些八卦杂志称为“动物界的刘翔”，广告不断，手头也有了不少积蓄。为了能够再赢兔子，乌龟不惜花下血本买了最先进的武器——“"小飞鸽"牌电动车。这辆车在有电的情况下能够以VT1 m/s的速度“飞驰”，可惜电池容量有限，每次充满电最多只能行驶C米的距离，以后就只能用脚来蹬了，乌龟用脚蹬时的速度为VT2 m/s。更过分的是，乌龟竟然在跑道上修建了很多很多（N个)的供电站，供自己给电动车充电。其中，每次充电需要花费T秒钟的时间。当然，乌龟经过一个充电站的时候可以选择去或不去充电。  
比赛马上开始了，兔子和带着充满电的电动车的乌龟并列站在起跑线上。你的任务就是写个程序，判断乌龟用最佳的方案进军时，能不能赢了一直以恒定速度奔跑的兔子。

Input

本题目包含多组测试，请处理到文件结束。每个测试包括四行：  
第一行是一个整数L代表跑道的总长度  
第二行包含三个整数N，C，T，分别表示充电站的个数，电动车冲满电以后能行驶的距离以及每次充电所需要的时间  
第三行也是三个整数VR，VT1，VT2，分别表示兔子跑步的速度，乌龟开电动车的速度，乌龟脚蹬电动车的速度  
第四行包含了N(N<=100)个整数p1,p2...pn,分别表示各个充电站离跑道起点的距离，其中0<p1<p2<...<pn<L  
其中每个数都在32位整型范围之内。

Output

当乌龟有可能赢的时候输出一行 “What a pity rabbit!"。否则输出一行"Good job,rabbit!";  
题目数据保证不会出现乌龟和兔子同时到达的情况。

Sample Input

100

3 20 5

5 8 2

10 40 60

100

3 60 5

5 8 2

10 40 60

Sample Output

Good job,rabbit!

What a pity rabbit!

**9. 母牛的故事**

**Problem Description**

有一头母牛，它每年年初生一头小母牛。每头小母牛从第四个年头开始，每年年初也生一头小母牛。请编程实现在第n年的时候，共有多少头母牛？

**Input**

输入数据由多个测试实例组成，每个测试实例占一行，包括一个整数n(0<n<55)，n的含义如题目中描述。  
n=0表示输入数据的结束，不做处理。

**Output**

对于每个测试实例，输出在第n年的时候母牛的数量。  
每个输出占一行。

**Sample Input**

2

4

5

0

**Sample Output**

2

4

6

**10. Brackets**

**Problem Description**

We give the following inductive definition of a “regular brackets” sequence:

* the empty sequence is a regular brackets sequence,
* if *s* is a regular brackets sequence, then (*s*) and [*s*] are regular brackets sequences, and
* if *a* and *b* are regular brackets sequences, then *ab* is a regular brackets sequence.
* no other sequence is a regular brackets sequence

For instance, all of the following character sequences are regular brackets sequences:

(), [], (()), ()[], ()[()]

while the following character sequences are not:

(, ], )(, ([)], ([(]

Given a brackets sequence of characters *a*1*a*2 … *an*, your goal is to find the length of the longest regular brackets sequence that is a subsequence of *s*. That is, you wish to find the largest *m* such that for indices *i*1, *i*2, …, *im* where 1 ≤ *i*1 < *i*2 < … < *im* ≤ *n*, *ai*1*ai*2 … *aim* is a regular brackets sequence.

Given the initial sequence ([([]])], the longest regular brackets subsequence is [([])].

**Input**

The input test file will contain multiple test cases. Each input test case consists of a single line containing only the characters (, ), [, and ]; each input test will have length between 1 and 100, inclusive. The end-of-file is marked by a line containing the word “end” and should not be processed.

**Output**

For each input case, the program should print the length of the longest possible regular brackets subsequence on a single line.

**Sample Input**

((()))

()()()

([]])

)[)(

([][][)

end

**Sample Output**

6

6

4

0

6