The 2013 ACM ASIA Preliminary Programming Contest Dhaka Site

Sponsored by IBM

Hosted by North South University
Dhaka, Bangladesh

16th November 2013
You get 13 Pages
8 Problems
&
300 Minutes

Problem A
Fallen Cookies!

As a guest of Fallen Land, you are invited to play a game. There are **N** cities in Fallen Land and **M** bidirectional roads connecting those cities. You know that every road connects exactly two different cities and if there is a road between cities **u** and **v** then you can go both ways from **u** to **v** or from **v** to **u**. Each of the **N** cities has some cookies. When you visit a city for the first time, you can pick all the cookies from that city. But there's another condition. There are some restrictions on number of times to visit a city. The welcome value of a city defines the maximum number of time you can visit that particular city. You can start the game from any of the **N** cities.

Find the maximum number of cookies you can collect and eat at the end of the day in the castle of Fallen Land.

Input
There will be multiple test cases. Input file starts with an integer **T** (**$0 < T <= 20$**) number of test cases.

Each of the following **T** test cases starts with two integers **N** and **M**. (**$1 <= N <= 10$**, **$0 <= M <= (N*(N-1)/2)$**), denoting the number of cities and number of roads in Fallen Land, respectively. In the next line, there are **N** integers, **$W_1, W_2, W_3, \ldots, W_N$**, where **$W_i$** indicates the welcome value of the **i**th city (**$0 < W_i < 3$**). The next line has **N** space separated integers, **$C_1, C_2, C_3, \ldots, C_N$** where **$C_i$** denotes the number of cookies in the **i**th city (**$0 <= C_i <= 1000$**). After that, there are **M** lines of input. On the **i**th line there is a pair of integers, **u_i** and **v_i** (**$1 <= u_i, v_i <= N, u_i != v_i$**) denoting there is a road between city **u_i** and city **v_i**.

Output
For each input, print the output in the format, **Case X: Y**. Here, **X** is the serial of the input and **Y** is the maximum number of cookies you can collect in Fallen Land following the rules maintained.

Sample Input	Output for Sample Input
2	Case 1: 3
3 2	Case 2: 2
1 1 2	
1 1 1	
1 2	
2 3	
3 1	
1 1 1	
1 1 2	
1 2	

Problem B
I am Dumb 4

One day our teacher asked: “Do you know how a lottery program works?” I promptly answered, “Isn’t it too easy? You just need to generate some unique random numbers within the range!” The teacher said, “The thing is not that simple, how can you ensure the distribution of winners is evenly distributed?” I said, “If I make sure that the distribution is even then how that can be random?” Anyway, the argument finished without any conclusion and I could not convince myself what he was telling me. Few years later I came across one lottery software and understood the whole thing. Suppose there will be 100 winners. Lottery Software will select 100 random numbers. Then it will divide the entire range into several sub-ranges of say 20. It will count for each sub-range how many winners are there. Then chi square test is performed to see how even the distribution is. There is some certain threshold value, if the test passes this limit the winners are accepted otherwise all the selected numbers are discarded and new winners are selected and the test is performed again. I still don’t agree with this process but you know I am dumb!

Now, given distribution of chocolates among 4 persons of your family (you, your mom, your dad and your sister) you are to tell me if the distribution is okay. A distribution will be okay if no person has received more chocolate than sum of other three members of family.

Input
First line of the input file is a positive integer **T** (**T <= 100**) which denotes the number of test cases. Each test case contains four numbers: **a**, **b**, **c** and **d** (**0 <= a, b, c, d <= 100**). These four numbers denote the number of chocolates received by each of your family members.

Output
For each test case print the case number followed by **Okay** or **Not Okay** depending on if the distribution is okay or not.

Sample input	**Output for Sample Input**

`2`
`10 1 2 1`
`1 2 1 1`
	`Case 1: Not Okay`
	`Case 2: Okay`

Template Code for C/C++:

```
#include<stdio.h>

int main()
{
	int cases, caseno, a, b, c, d;

	scanf("%d", &cases);
	for(caseno = 1; caseno <= cases; caseno++)
	{
		scanf("%d %d %d %d", &a, &b, &c, &d);
		printf("Case %d: ", caseno);
		if(....) //Replace the .... with proper logic
			printf("Okay\n");
		else
			printf("Not Okay\n");
	}

	return 0;
}
```

Problem C
Vocabulary Builder

Little Susan is studying Computer Science in her Bachelors programs. She wants to get into a good graduate school. But the first hurdle she faces is the dreadful GRE exam. She realizes that in order to ace the exam she has to build up her English vocabulary. She has written down a whole bunch of words. Now some of these words are related in that one is synonymous to another. Learning all the synonyms of a word helps Susan memorize new words easier. To test her word memorizing skills, Susan wants to be able to tell the word at a random position after the words are ordered based on their lexicographical ordering. Only catch is: she wants all the synonymous words to be put together.

For example, given the small list of words like:
dictate, conversation, advice, bidding, discussion, demand, suggestion, chat, talk, guidance and behest

Susan wants to order them as:

advice	guidance	suggestion	behest	bidding	demand	dictate	chat	conversation	discussion	talk
1	2	3	4	5	6	7	8	9	10	11

She knows that advice is synonymous to guidance and suggestion; behest is synonymous to bidding, demand and dictate; and chat is synonymous to conversation, discussion and talk. These word groups of synonyms need to be put together. Inside the groups themselves the synonyms need to be ordered in their lexicographical order. One group will be put before another if the first word in that group is lexicographically smaller than the other group's first word.

Your task is to help Susan in her exam preparation!

Input

The first line of the input will give the number of test cases, **T (1 <= T <= 10)**. Then **T** test cases follow. The first line of each test case will give you the number of words: **W (0 < W < 100,000)**. Each of these **W** words will come in a separate line. You can assume that all the words are unique, they contain only lower case letters from the alphabet ('a' to 'z') and they are all between **1** to **15** characters long. Following the word list you will be given the number of word pairings, **P** (**0 <= P < 10000**) as two words in a line separated by a single space. Each of these pairings indicate that these two words are synonymous. If word **x** is synonymous to **y** and **y** is synonymous to **z**, then **x** is also synonymous to **z**. Note that, each pair will contain two valid words listed before. And a pair may contain the same word twice and the same pair of words may come multiple times. After the synonymous words pairing, you will be given the queries description. First line of the query starts with the total number of queries, **Q** (**0 < Q <= W**). In each of the next **Q** lines, you will be given an integer, **N** (**0 < N <= W**) indicating the index of the word after they are ordered. In the test file there will be around **400,000** words, **60,000** word pairs and **2,000** queries.

Output

For each test case, you need to print the test case number **X** in the format “**Case X:**”. In the next **Q** lines for that test case you need to answer each query. For each query you need to print the word that is placed in the **N**th position after ordering them as Susan wants them ordered.

Sample Input	**Output for Sample Input**
1	Case 1:
11	advice
dictate	guidance
conversation	suggestion
advice	behest
bidding	bidding
discussion	demand
demand	dictate
suggestion	chat
chat	conversation
talk	discussion
guidance	talk
behest	
8	
dictate bidding	
guidance suggestion	
bidding behest	
guidance advice	
demand dictate	
conversation talk	
chat discussion	
conversation discussion	
11	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

Problem D
Greedy as Mr. Cracker

Mr. Cracker is a greedy rabbit. If he sees any carrot he becomes mad to get it. One day Association of Carrot Merchandiser arranged a contest for poor Tortoises. But Mr. Cracker does not care. Why should he care? The prize is **1** carrot! Cracker gets mad even if he sees others eating carrot. Even if it is half eaten carrot he is happy with it.

In the contest each of the contestants will be provided with four buckets of pearls. One bucket has **Red** pearls, one **Blue**, one **Green** and the last one **White** pearls. A contestant may color a White pearl with another color of his choice. When coloring will be finished they will give all these pearls to the judge Fox Shelly, head of the association. Shelly will then try to group the pearls. In each group there will be three pearls, one Red, one Blue and another Green. He will try to maximize number of groups for a contestant. The contestant who will produce maximum number of groupings will be the winner.

Fox Shelly is a friend of Rabbit Cracker. So he will give Rabbit Cracker the list of pearls beforehand. Unable to solve the problem (actually he even did not give it a try) Rabbit Cracker asked you to help him.

Input
First line of the test file contains a positive integer, number of test cases **T** (<= **100**). Hence follows **T** lines. Each line contains four non negative integers **R, G, B** and **W**. Total number of pearls will not be more than **1000,000**.

Output
For each test case print the case number and the maximum number of groupings possible.

Sample Input	Output for Sample Input

3
1 1 1 0
1 1 0 1
1 0 0 2
	Case 1: 1
	Case 2: 1
	Case 3: 1

Problem E
Chocolate Distribution

Given an **n x n** matrix, where each cell contains a chocolate. Jim is collecting all the chocolates and he wants to distribute them into sacks such that each sack contains exactly **m** chocolates (assume that he has unlimited number of sacks) with the exception that the last sack might have fewer chocolates.

Now he is wondering that if he decides to save **m** chocolates in each sack, what are the possible values of **n**, such that the last sack always has exactly **n** chocolates and it is fewer than other sacks (**n < m**)?

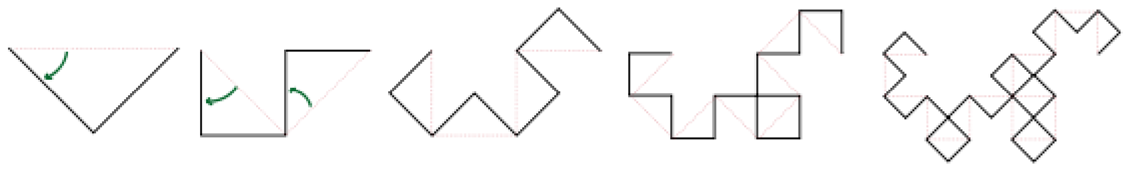
Input
The first of input contains a positive integer denoting the number of test cases **T** (**T <= 10000**).
Each case contains an integer **m (1 <= m < 2^{31})** in a single line.

Output
For each case of input, print one line containing the case number and the list of possible values of **n** in ascending order. Separate the integers by a single space. If there is no solution, print “**no solution**”.

Sample Input	**Output for Sample Input**
`4`	`Case 1: 5 6`
`10`	`Case 2: 6 10`
`15`	`Case 3: 9 16`
`24`	`Case 4: no solution`
`5`	

Problem F
Encoding The Hallway

Edward is now 21 years old. He has to appear in an exam to renew his "State Alchemist" title. This year the exam is arranged in a bit different way. There will be a long hallway. Each alchemist will enter the hallway from the left side and come out from the right side and he has to do this **n** times. During this tour they have to bend the hallway segments right-left alternatively. Let's describe the process by some pictures:



First time (First picture): Initially, the hallway is a straight line (soft line in the first picture). So alchemist will bend this segment to right side (he is going from left to right) like the hard line in the first picture above.

Second time (Second picture): Now he will find two segments in hallway (like soft line in picture). So he will bend the first hallway to right, second one to left (like the hard lines).

Third time (Third picture): Now he will find four segments in the hallway (like the soft lines) and he will bend them to Right, Left, Right and Left respectively.

And this goes on for fourth and fifth times in the picture.

Since Full Metal Alchemist Edward is so good, he did it perfectly. Now it is turn of the judges to check the bending if it is correct or not. The judge enters at the left end and comes out from the right end. But during his travel he notes down the turning, **R** for **Right** and **L** for **Left**. So if **n = 1**, then the judge would have noted down **L**. If **n = 2**, it would have been **LLR**. For **n = 4**, it would have been: **LLRLLRRLLLRRLRR**.

Since this string will grow exponentially with **n**, we are interested in a particular substring of the string.

Input

First line of the test file contains a positive integer **T** denoting number of test cases (**T <= 100**). Hence follows **T** lines, each containing three integers: **n a b**. **n** is the number of times Edward has passed through the hallway; **a** and **b** are indices in the string (**1 <= n <= 1000**). We will consider the string to be **0** indexed and **0 <= a <= b < length of the string written by the judge**. (**0 <= b – a <= 100, b <= 10^{18}**)

Output

For each test case output the case number and the substring starting from **a**'th position and ending at **b**'th position. Consider the starting and ending position as inclusive bound.

Sample Input	Output for Sample Input
`2`	`Case 1: L`
`1 0 0`	`Case 2: LRRLL`
`4 4 8`	

Problem G

Hiring Optimally

Entrepreneurship should always be encouraged. Rather than working in a service, everyone should try to have one's own business. It creates more opportunities and help people finding their own way. Entrepreneurs are the one who shapes the world.

Yu wants to be an entrepreneur. With a lot of enthusiasm, energy and hope, Yu is starting his new business. But soon Yu realizes, it's not all fun. Specially, taking interview of a lot of programmers can be a hectic task. So Yu decides, he will take interview once and choose the best and brightest one among them. A lot of discussing, processing, calling candidates to let them know whether or not they are hired etc. are really not his favorite tasks either. So he decides, he will let the candidate know the result of the interview on the spot immediately after their interview is finished. Means, after finishing the interview, a candidate will leave the room knowing whether he is accepted for the job or rejected. If rejected, that person will not be called back again. If accepted, no more interviews will be taken after that candidate.

Yu is an excellent judge of programmers. He can always rank the already interviewed candidates according to their merit but can't know their actual merit. He knows the number of candidates, **N**, lined up for the interview. He also knows that there are no two candidates who are equally qualified. Let **N = 5**. An actual ordering of candidates can be **3, 1, 2, 5, 4** (**1** being the worst and **5** being the best candidate according to merit). After interviewing with the first three candidates, Yu will realize the first candidate is the best one so far, the second candidate is the worst one and the third one is between them. But he will have no idea about their actual merit (that the first one is actually third in merit among all the 5 candidates).

Problem is, Yu has no idea about the order of candidates in which they may appear in the interview. So it is possible that he will reject the best (overall best, among **N** candidates but not known to Yu then) candidate in hope for a better one. He may also accept a candidate without even meeting the best one (the best candidate comes after the accepted candidate). In both cases, he ends up hiring a candidate who is not the best (Yu's nightmare).

Being a programmer himself, Yu comes up with a strategy. It's called **Strategy(K)**, where **K** is an integer between **0** and **N – 1**. It works following the rules mentioned below:

1. Whatever happens, Yu will always reject the first **K** candidates he interviews.
2. After that, Yu will accept the first candidate, who is better than all of the first **K** candidates.
3. If Yu cannot find a one who is better than the first **K** candidates, then he will accept the last or **N**th candidate.
4. If **K** is **0**, that means Yu will accept the first candidate.

For example, if **N = 5** and **K = 2** and the actual ordering of the candidates are **3, 1, 2, 5, 4** (**1** being the worst and **5** being the best candidate according to merit), Yu will reject the first and second candidate (but he will note their merit) according to rule 1. He will reject the third candidate because he is not better than all of the first two (**K = 2**). He will accept the fourth candidate because he is better than the first two. In this scenario, Yu will hire the overall best. If the actual ordering is **3, 1, 2, 4, 5** (**N** and **K** unchanged) then Yu will still hire the fourth candidate but who is not the overall best (second best, but still not the best).

So you can see, it’s not sure whether Yu will get the best candidate or not, following **Strategy(K)**. **Prob(K)** denotes the probability that Yu will hire the best candidate following **Strategy(K)**. Now Yu need to find the value of **Prob(K)** given the value of **K**.

Input

First line of input contains a single integer, **T (T <= 500)**, the number of test cases. **T** lines follows. Each consists of an integer **N (1 <= N <= 10^8)**, total number of candidates in the interview, and an integer **K (0 <= K < N)**, for which you need to find **Prob(K)**.

Output

For each case print one line “**Case X: P**” (without the quotes), where **X** is the case number and **P** is the value of **Prob(K)**. Print the answer rounded to four digits after decimal point. You can be assured that the answer will not change even after adding **1e-7**.

Sample Input	**Output for Sample Input**

`3`
`3 1`
`5 2`
`100 3`
	`Case 1: 0.5000`
	`Case 2: 0.4333`
	`Case 3: 0.1103`

Explanation

For case 1, there are in total 3! = 6 possible orderings.
For Strategy(1) => 132, 231, 213 will give Yu the best candidate. So, Prob(1) = 3/6 = 0.5

Problem H
Large Factorials

Anannya just learned about factorial and loved it very much. In case you don't know about factorial, she can define the factorial function of a number **n** as:

$$factorial(n) = \begin{cases} 1, & n = 0 \\ n * factorial(n-1), & n \geq 1 \end{cases}$$

We can define it more easily:

$$factorial(n) = \begin{cases} 1, & n = 0 \\ 1 * 2 * 3 * \ldots * n, & n \geq 1 \end{cases}$$

Now Anannya wants to find factorial of large numbers. But she does not know about **64**-bit or big integer, she recognize only **32**-bit integers. So she wants to calculate ***factorial(n)*** **modulo** $\mathbf{2^{32}}$, given **n**.

Input
Input starts with a positive integer **T (T <= 10000)**, number of test cases to follow. Each test case contains only a single integer **n (0 <= n <= 10**^{9}**)** on separate lines.

Output
For each test case, print the test case number and ***factorial(n)*** **modulo** $\mathbf{2^{32}}$.

Sample Input	Output for Sample Input
3 0 1 5	Case 1: 1 Case 2: 1 Case 3: 120