VCU School of Engineering
2014 Computer Science Programming Contest

Welcome to the 2014 Programming Contest. Before you start the contest, please be aware of the following:

1. There are ten (10) problems in the packet, using letters A-J. These problems are NOT sorted by difficulty. When a team's solution is judged correct, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	Ring	Dark Blue
B	Warp Gates	Light Blue
C	Boxes	Gold
D	Perfect Number	Green
E	One-Time Pad Encryption	White
F	Hubris of Olympic Proportions	Red
G	Goldbach's Conjecture	Orange
H	Ties	Yellow
I	Smart	Black
J	Cloud Control	Pink

2. All solutions must read from standard input (System.in) and write to standard output (System.out).

3. Solutions for problems submitted for judging are called runs. Each run will be judged. Runs for each particular problem will be judged in the order in which they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for B followed by a run for C, but receive the response for C first. DO NOT request clarifications on when a response will be returned. If you have not received a response for a run within 10 minutes of submitting it, **you may ask**

the lab monitor to send a runner to the judge to determine the cause of the delay. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.

The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	**Explanation**
Yes	Your submission has been judged correct.
Wrong Answer	Your submission generated output that is not correct.
Output Format Error	Your submission's output is not in the correct format, is misspelled, or did not produce all of the required output.
Excessive Output	Your submission generated output in addition to what is required.
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	You submission did not solve the judges' test data within **2 minutes**.

4. A team's score is based on the number of problems they solve and penalty points. The penalty points reflect the time required to solve a problem correctly and the number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved. Teams are ranked by the number of problems solved. Ties are resolved in favor of the team with the fewest penalty points.

5. This problem set contains sample input and output for each problem. However, you may be assured that the judges will test your submission against several other more complex datasets, which will not be revealed. Before submitting your run you should design other input sets to fully test your program. Should you receive an incorrect judgment, you are advised to consider what other datasets you could design to further evaluate your program.

6. In the event that you think a problem statement is ambiguous, you may request a clarification. Read the problem carefully before requesting a clarification. If the judges believe that the problem statement is sufficiently clear, you will receive the response,

“The problem statement is sufficient, no clarification is necessary.” If you receive this response, you should read the problem description more carefully. If you still think there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found. If the problem statement is ambiguous in specifying the correct output for a particular input, please include that input data in the clarification request.

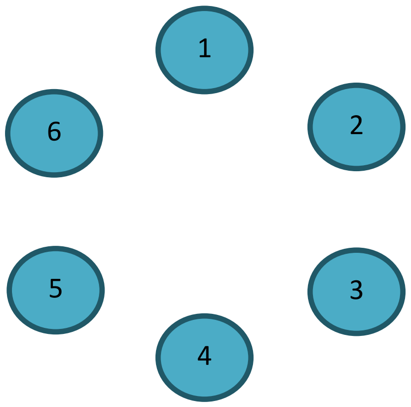
Additionally, you may submit a clarification request asking for the correct output for input you provide. The judges will seek to respond to these requests with the correct output. These clarification requests will be answered only when no clarifications regarding ambiguity are pending. The judges reserve the right to suspend responding to these requests during the contest.
If a general clarification, including output for a given input, is issued during the contest, it will be broadcast to all teams.

7. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.

8. Good luck and HAVE FUN!!

Problem A
Ring

N people are playing a game. They sit in a circle with N seats. The following example shows a circle with 6 seats.



Starting from the person sitting on seat “1”, they count off clockwise by three. The person who has “three” loses and leaves the circle. Then, the rest continue playing the game by starting from the person who is the clockwise next to the leaving one until there is only one left. The one left is the winner.
In the circle above, the first person to be eliminated is person seating at seat “3”, then “6”, then “4”, until only person seating at seat “1” is left.

Input
Single integer 1<N<=100000 that indicates the number game participants.

Output
Single integer: the winner’s seat number.

Sample Input
6

Sample Output
1

Problem B
Warp Gates

The Interdimensional Travel Department is developing a plan to promote travel between dimensions of the known Universe. They have decided to construct warp gates between dimensions but want to keep costs down. The latest warp gate technology is to be deployed with bi-directional mass transportation capabilities, each connecting two dimensions. It is your task to advise the Department which warp gates to construct to make travel between all (n) dimensions possible with minimal construction cost.

Input

The first row of the input contains two integers:

- the first integer (n; 2<=n<=1000) represents the number of dimensions that must be connected.
- the second positive integer (m; 1<=m<=10000) represents the number of possible warpgates being considered.

Following are (m) lines with three integers on each line:

- The first integer identifies the first dimension being connected by the warpgate.
- The second integer identifies the second dimension being connected by the warpgate.
- The third integer (between 1 and 100) identifies the cost of constructing this warpgate.

All values are separated by spaces. Dimensions are identified as unique integers {0...n-1}. Since warpgates are bi-directional, they are guaranteed to be listed only one time, but travel may occur in both directions.

Output

The output of your program will be the minimum cost for connecting all (n) dimensions.

Sample Input

```
5 6
0 1 2
0 2 3
0 3 5
1 4 4
2 3 6
2 4 1
```

Sample Output

```
11
```

Problem C
Boxes

George wants to give a surprise to his girlfriend by wrapping a small gift into a bunch of boxes. For example, he can put the gift into a small box first and then put the small box into a larger box, and so on. Suppose a box can fit into another box only if its dimensions are strictly smaller than the other one. George plans to use as many boxes as possible. Given a bunch of *n* boxes and their dimensions, denoted by {L1, W1, H1}, {L2, W2, H2}, … , {Ln, Wn, Hn}, how many boxes can he use at most to wrap the gift into?
Assume the gift can fit into any box and the boxes can be rotated. Also the dimensions of all boxes are represented by integers. We assume box A fits into box B if and only if A's {L, W, H} are strictly smaller than B's. However, boxes can be rotated so their L, W, H can switch. For example, box {1,2,3} can fit into {4,3,2}; box {1,2,3} cannot fit into {2,2,4}.

Input
First input line consists of the number of boxes, *n*<=100.
Each of the subsequent *n* input lines consists of three positive integers between 1 and 10000, representing the size of the box along the L, W and H dimensions.

Output
One positive integer S: the maximum number of boxes that can be used.

Sample Input
```
4
1 2 1
2 3 4
5 4 3
2 5 2
```

Sample Output
```
3
```

Problem D
Perfect Number

A positive integer is termed a perfect number if the sum of all of its divisors less than itself is equal to the number.

Write a program that identifies if an input is a perfect number or not.

Input
First input line consists of a number of integers to follow, C<=100.
Each subsequent input line consists of a positive integer between 2 and 10000.

Output
For each input positive integer N, you should output all positive divisors less than N, in a single line in an increasing order. Next, in the same line, you are to output YES if N is a perfect number and NO otherwise.

Sample Input
```
2
6
34
```

Sample Output
```
1 2 3 YES
1 2 17 NO
```

Problem E
One-Time Pad Encryption

Alice and Bob went to a cryptography seminar, and learned a trick to protect their personal communication, named ``One-Time Pad''. The main idea is that, given a sequence of numbers (a pad), any message (plaintext) can be proven to be securely encrypted into a new message (ciphertext) if each plaintext character is encoded into a ciphertext character by a number from the pad, if each number from the pad is only used once.

Alice then decides to use the one-time pad to encrypt her messages for Bob by shifting each letter in the plaintext message by k positions later in the alphabet, where k is determined by the one-time pad. Shifts occur in alphabetical order, between the letters ``a'' and ``z''. If a letter is shifted past ``z'', it starts back at ``a'' and continues shifting. For example, shifting a letter by k = 25, the plaintext ``c'' is transformed into a ciphertext ``b'', and similarly, shifting by k= 2, the plaintext ``z'' is shifted to ``b''.

Input
The input consists of four parts:
1) start with the size of the one-time pad;
2) then followed by a sequence of numbers for the pad;
3) then followed by a series of words for the plaintext;
4) and finally terminated by a line containing only -1.

You may assume that the maximum size of the pad is 100 numbers, all numbers in the pad are non-negative integers below 1000, and that all input plaintext words will be lowercase letters.

Output
For each plaintext to be encrypted, output a line containing the corresponding ciphertext.

Sample Input
```
5
1 2 3 4 5
bb zzz
-1
```

Sample Output
```
cd cde
```

Problem F
Hubris of Olympic Proportions

Almost every contestant who makes it to the Olympic Games aims to perform above average. Can they all reach that goal? You need to conduct a study.

Input
The first line of standard input contains an integer C, the number of test cases. C data sets follow. Each data set begins with an integer, N, the number of sportsmen and sportswomen (1 <= N <= 1000). N integers follow, separated by spaces, each giving the final score of one contestant (normalized to be an integer between 0 and 100; the gold medalist gets a 100).

Output
For each case you are to output a line giving the percentage of contestants whose score is above average, rounded to the closest integer.

Sample Input
```
5
5 50 50 70 80 100
7 100 95 90 80 70 60 50
3 70 90 80
3 70 90 81
9 100 99 98 97 96 95 94 93 91
```

Sample Output
```
40
57
33
67
56
```

Problem G
Goldbach's Conjecture

The famous Goldbach's conjecture, a well-known unsolved problem in number theory, states that every even integer greater than 2 can be written as a sum of two prime numbers. For nearly 300 years, even the best mathematician has not been able to crack it.

Let $G(N)$ be the number of different ways to represent a number *N* as a sum of two prime numbers.
The Goldbach's conjecture can be stated as follows: $G(N) > 0$ for all even $N \geq 4$.
For example, *G*(4) = 1 and *G*(18) = 2.

Given an integer N, your task is to write a program to compute *F*(*N*), the sum of *G* for all even numbers from 4 up to $2N$:

$$F(N) = G(4) + \ldots + G(2N)$$

<u>Input</u>
The first line of the input file contains a number $1 \leq K \leq 20$. The following K lines contains K integers $N_1, N_2, \ldots, N_k$, one per line, where $(3 \leq N_i \leq 500000)$ for $i = 1..K$.

<u>Output</u>
For each N_i output on a separate line the value of $F(N_i)$.

<u>Sample Input</u>
```
2
9
7
```

<u>Sample Output</u>
```
12
8
```

Problem H
Ties

Ana and Voyo are practicing hard for the new programming contest season. They hold many private contests where only the two of them compete against each other. They have almost identical knowledge and skills, so they are ending up being many times in ties, in terms of both the number of problems solved and in time penalty! To break the tie, Ana and Voyo invented a tie breaker technique called sequence breaking! The technique goes as follows:

1- Generate a random integer N >= 2.
2- Generate a sequence of N random integers.
3- If N = 2 go to step 6.
4- Fold the sequence by adding the Nth element to the first, the N-1th element to the second and so on. If N is odd then the middle element is added to itself, figure 1 illustrates the folding process.
5- Set N = ceil (N/2) and go to step 3.
6- The sequence now contains two numbers, if the first one is greater than the second number, Ana wins, otherwise Voyo wins. As an old fashioned gentleman Voyo agrees that in the, rare indeed, case of a two numbers being equal Ana wins.

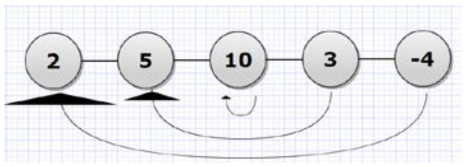


Figure 1.a Before Folding

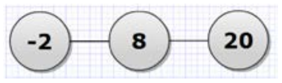


Figure 1.b After one step of folding

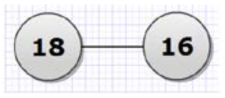


Figure 1.c After two steps of folding, Ana wins!

In this problem you are given the sequence of N integers and are asked to determine the contest winner using the tie breaker technique described above.

Input
The first line contains T (1 <= T <= 100), the number of test cases. The first line of each test case contains an integer (2 <= N <= 100), the number of elements of the sequence. The next line contains N space separated integers. The sum of any subset of the numbers fit in a 32 bit signed integer.

Output
For each test case print the name of the winner. Follow strictly the output format below.

Sample Input
```
2
5
2 5 10 3 -4
3
5 4 -3
```

Sample Output
```
Ana
Voyo
```

Problem I
Smart

VCU basketball coach knows games are won using brains, not just muscle. In practice games he keeps the score of each side to himself, and instead shows on the board the sum of the two scores, and the absolute difference between the two scores. The players need to solve this riddle in real-time to keep track if their side is winning. Can you write a program that solves this riddle?

Input
The first line of input contains n, the number of test cases, (1<=n<=1000). n lines follow, each representing a test case. Each test case gives s and d, non-negative integers representing the sum and (absolute) difference between the two scores. Both s and d are below 500.

Output
For each test case, output a line giving the two final scores, largest first. If there are no such scores, output a line containing "impossible".

Sample Input
```
2
40 20
20 40
```

Sample Output
```
30 10
impossible
```

Problem J
Cloud Control

A company runs a cloud with N servers (numbered 0, …, N-1) and offers users an easy to use computing platform with a predefined set of J subroutines (numbered 0, …, J-1). Users specify a program as a sequence of T tasks, each task involving one call to a subroutine that processes the input data in some way. That is, a program is a sequence of T numbers each from (0, …, J-1). The tasks have to be executed in order, that is, processing of a task starts only after the task preceding it in the sequence finishes.

Subroutines (0, …, J-1) are all different – some need fast disk access, some a fast CPU, other a fast graphics card. To handle this, the servers in the cloud are diverse. The engineers have measured how long it takes to run each subroutine on each server. In this way they obtained the data processing time matrix, DPT, where DPT[A][x] captures how long it takes to execute subroutine x on server A. The engineers also have quantified how long it takes to move the data from each server to each other server, that is, they have the matrix of data transfer times, DTT.

Now the company needs a cloud manager: a program that takes the sequence of T tasks a user submits, then for each task sees which subroutine is used and decides on which of the N servers that task should be executed. The goal is to minimize the total time from the submission of the series of tasks to the completion of the last of the tasks. Of course, if the first task is executed on server A, and the second on server B, the data being processed will need to be moved from A to B. So, the subroutine on B will not start right at the moment the subroutine on A finishes – data transfer will cause a delay, equal to what is specified in the data transfer matrix, in DTT[A][B].

Your job is to write the cloud manager. For an input sequence of T tasks, given the matrix of data processing times and the matrix of data transfer times, figure out the sequence of servers for the tasks that results in shortest total time needed to complete all the tasks. Assume that prior to execution of the first task, data reside on server 0, so if the first task is executed somewhere else, data need to be transferred first. The final result of processing, the results of the T-th task, is to be left at the server where that task was processed.

For example, the sample input below defines a cloud with 3 servers (0, 1, 2) and two subroutines (0, 1). With the processing and transfer times given below, the optimal way of solving a series of 3 tasks involving, in order, subroutines 0, 1, 0, is to move data from server 0 to server 1, execute subroutine 0 at server 1, move data from server 1 to server 2, execute subroutine 1 there, then move data from server 2 to server 0, and finally execute subroutine 0 there.

Input

- Line 1: integer N<=100 – the number of servers
- Line 2: integer J<=50 – the number of available distinct subroutines
- Line 3: integer T<=100 – the number of task in a sequence
- Then, N lines follow, each containing N integers. Together, they specify the N by N matrix of data transfer times DTT from server to server. The matrix element in row i and column j denotes the transfer time from server i to server j. The time is 0 if i equals to j, and is an integer between 1 and 10000 otherwise.
- Next, N lines follow, each containing J integers. Together, they specify the N by J matrix of data processing times, DPT. The matrix element in row i and column j denotes the processing time of subroutine j on server i. Each processing time is an integer between 1 and 10000.
- Finally, a single line of T integers follows describing the user's sequence of tasks, that is, the subroutines to be executed and the order in which they are to be executed. Each integer is a number between 0 and J-1.

Output

A single integer: the shortest time in which the series of tasks can be completed on the cloud.

Sample Input

```
3
2
3
0 10 100
300 0 20
30 200 0
15 500
3 80
1000 5
0 1 0
```

Sample Output

```
83
```