University Interscholastic League

Computer Science Competition

2013 Invitational A Programming Problem Set

DO NOT OPEN THIS PACKET UNTIL INSTRUCTED TO BEGIN!

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points. Incorrect submissions receive a deduction of 5 points, but may be reworked and resubmitted. Deductions are only included in the team score for problems that are ultimately solved correctly.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

II. Names of Problems

Number	Name
Problem 1	Ciphertext
Problem 2	Circular Prime
Problem 3	Dominoes
Problem 4	Heads Up
Problem 5	Hindsight is 20/20
Problem 6	Monotone Submatrix
Problem 7	Pseudorandom Numbers
Problem 8	Sandbox
Problem 9	Slot Machine
Problem 10	Summer Job
Problem 11	Text Messages
Problem 12	Upward Bound

1. Ciphertext

Program Name: Ciphertext.java **Input File: ciphertext.dat**

Sally and Nancy have decided to write each other in a special code. Being middle school students, they are not very clever. They decide to use a straight replacement method, commonly called a substitution cipher. In a substitution cipher, a list of the 26 distinct alphabetic letters in a list, called ciphertext, is used to replace the 26 letters of plain text by straight substitution. Non-alphabetic characters are not replaced.

They have decided on the string "ASDFGHJKLQETUOWRYIPZCBMNVX" as the list to use as their ciphertext, where each letter in the ciphertext replaces the corresponding letter from the alphabet "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

You are to write a program that will take a note that Nancy has written in ciphertext and translate it to plain text so Sally can read it. Using the string above, you would replace each letter A in their ciphertext letter with the letter A, each letter S with a B, each letter D with a C, each letter F with a D, and so forth until all of the ciphertext characters in their letter have been replaced with the corresponding letter in the alphabet so they can read the letter from their friend.

Input

The first line of input will contain a single integer n that indicates the number of coded sentences to follow. Each of the following n lines will contain a single coded note composed of ASCII characters from the keyboard. All alphabetic letters will be uppercase.

Output

For each coded note input, you will print the decoded note.

Example Input File

```
3
ZKG CLT PZAZG UGGZ LP LO UAV. L MLTT PGG VWC LO ACPZLO.
QWKO LP LO UV PZCFV JIWCR. KG EGGRP UG HWDCPGF WO UV PZCFLGP.
UV UWZKGI LP UV SGPZ HILGOF. PKG COFGIPZAOFP KWM L ZKLOE.
```

Example Output to Screen

```
THE UIL STATE MEET IS IN MAY. I WILL SEE YOU IN AUSTIN.
JOHN IS IN MY STUDY GROUP. HE KEEPS ME FOCUSED ON MY STUDIES.
MY MOTHER IS MY BEST FRIEND. SHE UNDERSTANDS HOW I THINK.
```

2. Circular Prime

Program Name: CirPrime.java Input File: cirprime.dat

A circular prime number is an integer that is a prime number for each rotation of the digits that form the integer. A rotation is created when the right-most (or last) digit in the number is moved to become the left-most (or first) digit in the number while all the other digits shift over to the right to make room but maintain their order. For example, the integer 719 is a circular prime because each of its three rotations, 719, 971, and 197, is a prime number.

You are to write a program that will determine if a given number is a circular prime.

Input
The first line of input will contain a single integer `n` that indicates the number of integers to follow. Each of the following `n` lines will contain a single positive integer `c` ($100 < c < 1{,}000{,}000$).

Output
For each integer input and on a separate line, you will print `YES` if it is a circular prime and print `NO` if it is not.

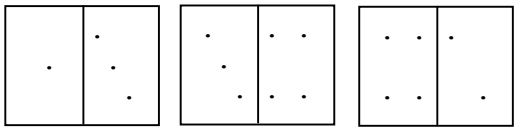
Example Input File
```
3
719
19937
1079
```

Example Output to Screen
```
YES
YES
NO
```

3. Dominoes

Program Name: Dominoes.java **Input File: dominoes.dat**

Lester is trying to teach his daughter Amy about numbers by playing a game using dominoes. Their box of dominoes contains 21 dominoes, each of which has two "ends". There are six "ones" which means one end is a one and the other end is a number one through six. There are five “twos” (after eliminating the duplicate 2-1 vs. 1-2). Similarly, there are four "threes", three "fours", two "fives" and one "six" for a total of 21 dominoes. For the game they are playing, they randomly select seven dominoes from the box and try to form a line by putting matching ends together. For example, if they have the dominoes 3-1, 4-3 and 4-2, they could form the line: `1-3 3-4 4-2`.



Input

The first line of input will contain a single integer `n` that indicates the number of games they will play. Each of the following `n` lines will contain seven dominoes, each in the form `x-y` where `x` and `y` are the values for each end of one domino. Each of dominoes will be separated by a space.

Output

For each game, you will output:

`YES` if all seven dominoes can be placed end –to-end as described above, or
`NO` if it is not possible.

Example Input File

```
2
1-2 4-3 4-4 2-5 4-2 1-4 3-5
4-5 6-6 1-6 5-6 3-4 2-1 4-1
```

Example Output to Screen

```
YES
NO
```

4. Heads Up

Program Name: HeadsUp.java Input File: headsup.dat

James is writing a game app that involves random binary outcomes of a coin flip. He would like for the outcome of each event in his app to always have a success rate of between 45 and 55 percent, inclusive. An outcome is considered to be a success if the coin lands on heads.

For each event, you will need to construct an object of the type `java.util.Random`. This class allows you to specify the seed for the random number generator. For a given seed, the order of the random numbers is always the same. Then, for each event, you will determine if the given seed and the number of trials for your `Random` object will generate enough heads for the outcome to be considered a success. For this problem assume that 0 stands for heads and 1 stands for tails.

Input

The first line of input will contain a single integer `n` that indicates the number of events to follow. Each of the following `n` lines will contain a `Long`, which is the seed, followed by a space and an integer, which is the number of trials for that event.

Output

For each event, you will print `YES` if the outcome of that event is a success or `NO` if the outcome is not a success.

Note: Truncate the percent before determining if the outcome is a success or not.

Example Input File

```
4
4567433466 100
1234677333 200
1234677333 20
63345678212122 500
```

Example Output to Screen

```
YES
NO
NO
YES
```

Note: these are the random numbers generated for the first three seeds in the input file:

```
4567433466
1 0 0 1 0 0 1 0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0
0 0 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 1 1 0 1 1 0 0 0 1 0
0 1 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 1 1 0 1 1
1234677333
1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1
0 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0
0 1 1 0 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
1 1 0 0 1 0 1 0 1 1 0 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 0 1 1 1 0 1 1 1 0
1 1 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0
1 1 0 0 1
1234677333
1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 1 0
```

5. Hindsight is 20/20

Program Name: Hindsight.java **Input File: hindsight.dat**

Everyone knows that hindsight is 20/20. Ethan is studying some stock trends in hopes of gaining some insight into how to buy and sell stocks to make a maximum profit. Ideally, a person's maximum profit is determined by buying a stock on the day the price of the stock is as low as possible and then selling that stock when the price is as high as possible. However, if the price of the stock never increases after he buys the stock, his profit would be zero, or even worse if the price decreases.

You will be given a list of prices of a particular stock every day for two weeks. You will write a program that will print the maximum amount Ethan could earn in profit for the given two weeks. If he could not sell for a profit, he will not sell the stock at all, so his profit would be zero (0).

Input

The first line of input will contain a single integer `n` that indicates the number of stocks to follow. Each of the following `n` lines will contain 14 integers separated by a space. Each integer will represent the price of the stock on days 1 through 14.

Output

For each stock, you will print the maximum amount Ethan could earn in profit for the given two weeks or zero(0) if he could not make a profit.

Example Input File

```
3
10 15 12 16 12 15 18 12 10 8 4 13 12 8
5 8 12 13 4 6 12 9 8 7 12 15 12 16
12 12 12 11 11 11 10 10 10 10 9 9 8 7
```

Example Output to Screen

```
9
12
0
```

6. Monotone Submatrix

Program Name: Monotone.java **Input File: monotone.dat**

A matrix is said to be monotone if all elements in each row are strictly non-decreasing when read from left to right and the elements in each column are strictly non-decreasing when read from top to bottom.

A submatrix of a given matrix is a rectangular region of the matrix containing only consecutive rows and columns. The size of a matrix or submatrix is the number of elements it contains. Given a matrix, you will determine its largest monotone submatrix.

Input

The first line of input will contain a single integer `n` that indicates the number of matrices to follow. For each matrix, the first line will contain two integers `r c` that denote the number of rows and columns respectively in the matrix. The following `r` lines will contain `c` positive integers less than 100, each separated by a space.

Output

For each matrix, you will print the largest monotone submatrix contained in the given matrix. Print a blank line after each of the submatrices. Assume that each matrix will have only one largest submatrix. Assume also that there is a minimum size for the submatrix of 2 rows and 2 columns.

Notes:

- A space at the end of each line is optional.
- A blank line at the end of the last submatrix is optional.

Example Input File

```
2
5 5
17 12 13 15 20
12 13 16 17 22
8 8 16 18 23
7 9 17 19 25
9 9 9 20 25
5 7
7 8 9 10 12 12 15
9 11 14 10 19 2 45
12 13 18 2 25 45 66
11 13 18 25 28 29 77
14 16 19 29 35 44 88
```

Example Output to Screen

```
13 15 20
16 17 22
16 18 23
17 19 25

11 13 18 25 28 29 77
14 16 19 29 35 44 88
```

7. Pseudorandom Numbers

Program Name: Pseudo.java **Input File: pseudo.dat**

Bryan wants to generate random values to use in an app that he is writing but he is concerned about both speed and memory. He has decided to use a pseudorandom number generator, Linear Congruential Generator (LCG), which is fast and uses little memory. A plus is that an LCG is easy to implement; a minus is that it should only be used in certain types of applications.

When using an LCG, a new number `N` is created using these steps:

1. multiply the previous number `P`, also called the seed, by a given constant `C`
2. increment the product in step 1 by another constant `I`
3. mod the result in step 2 by another constant `M`

The formula is `N=(P*C+I) (mod M)`. This formula will generate at most `M` distinct pseudorandom numbers before repeating the sequence of numbers again. Hence, an LCG is considered to be cyclic.

You are to write a program that will print, in the order they are created, all of the pseudorandom numbers created until the first cycle is complete.

Input

The first line of input will contain a single integer `n` that indicates the number LCGs that you are to create. Each of the following `n` lines will contain 4 integers `P C I M` as defined above. Each of items will be separated by a space.

Output

On a single line, separated by a space, and in the order they are created, you will print all of the pseudorandom numbers created until the first cycle is complete.

Note: A space is optional at the end of each line.

Example Input File

```
3
4 7 5 12
5 5 3 7
12 13 15 16
```

Example Output to Screen

```
9 8 1 0 5 4
0 3 4 2 6 5
11 14 5 0 15 2 9 4 3 6 13 8 7 10 1 12
```

8. Sandbox

Program Name: Sandbox.java **Input File: sandbox.dat**

George owns a landscaping business. One of the things customers buy is sand for their child's sandbox which he sells in bags, each of which contains 2 cubic feet of sand. He needs you to write an app that will tell the customer the number of bags he needs to buy to fill a sandbox given the length and width of the sandbox in feet and the depth in inches.

Input

The first line of input will contain a single integer `n` that indicates the number of sandboxes to follow. Each of the following `n` lines will contain three positive integers `l w d`, where `l` is the length in feet of the sandbox, `w` is the width in feet of the sandbox, and `d` is the depth in inches of the sandbox.

Output

For each sandbox and on a single line, you will print the number of bags of sand the customer will need to buy to fill the sandbox.

Example Input File

```
4
3 4 8
8 10 10
12 7 12
4 3 14
```

Example Output to Screen

```
4
34
42
7
```

9. Slot Machine

Program Name: Slots.java Input File: slots.dat

Your team has been assigned to write a slot machine game. Others in the group are going to write the user interface for the slot machine and you have been selected to write the part of the game that decides whether or not a person is a winner.

Your slot machine has three identical wheels with three objects on each wheel. The machine randomly selects one object from each of the three wheels. If the three objects selected are the same, the person is a winner.

For each game, you will need to construct one object of the type `java.util.Random`. This class allows you to specify the seed for the random number generator. For a given seed, the order of the random numbers is always the same.

Input

The first line of input will contain a single integer `n` that indicates the number of games in your test class. Each of the following `n` lines will contain one `long` integer that you are to use as a seed for that game.

Output

For each game, you will print `WINNER` if the three wheels match or `NOT WINNER` if they do not match.

Example Input File

```
3
342345456
564555245
4354564
```

Example Output to Screen

```
NOT WINNER
NOT WINNER
WINNER
```

Note: these are the random numbers generated for each seed in the input file:

```
342345456
2 2 1
564555245
2 1 1
4354564
1 1 1
```

10. Summer Job

Program Name: SummerJob.java **Input File: summerjob.dat**

James is hoping to get a summer job flipping burgers at the local Hamburger King. His starting salary will be $7.25 per hour. However, there are certain incentives to encourage him to work extra hours. This is the way his pay will be figured per day:

- $7.25 for the first 6 hours he works per day, Monday through Friday.
- $8.25 for the next 2 hours he works per day, Monday through Friday.
- $11.25 for each hour he works over 8 hours in a day, Monday through Friday.
- On Saturday and Sunday, his pay is $11.25 per hour regardless of the number of hours he works.
- He will receive an additional $4 per hour bonus for each hour he works over 40 hours in a week.

You are to write a program that will compute the amount of money James would receive for a week's work given the number of hours James plans to work each day.

Input

The first line of input will contain a single integer `n` that indicates the number of weeks James plans to work. Each of the following `n` lines will contain seven positive doubles less than 12 indicating the number of hours, rounded to the nearest half-hour, that James plans to work each day in order, Sunday through Saturday. Each of the items will be separated by a space.

Output

For each week input, you will print on a separate line, the amount of money James would make that week in the form `$ddd.cc`, where `d` is dollars with no leading zeros and `c` is cents to the nearest penny.

Example Input File

```
4
0 5.5 7 4.5 9 10 4
11 5 8.5 10 3.5 0 0
4 6 6 7.5 4 7 11
9 9 10.5 8.5 6 9 0
```

Example Output to Screen

```
$323.00
$333.50
$414.38
$489.00
```

11. Text Messages

Program Name: Texts.java Input File: texts.dat

James wants to automate his text messages to his friends. You will write a program that will take the message that James wants to send, remove all of the uppercase and lowercase vowels, and then print the final message.

Input
There will be an unknown number of lines, each of which will contain one or more sentences.

Output
For each line input, you will print the sentences with all vowels removed.

Note: Vowels are the uppercase and lowercase letters: `A`, `E`, `I`, `O`, and `U`.

Example Input File
```
Computer science class is the most fun class in high school. Only the best
students take it.
There is nothing I would rather do than compete in a Computer Science contest
on Saturday morning.
```

Example Output to Screen
```
Cmptr scnc clss s th mst fn clss n hgh schl. nly th bst stdnts tk t.
Thr s nthng  wld rthr d thn cmpt n  Cmptr Scnc cntst n Strdy mrnng.
```

Note: Both of the lines in the input file stretch across two lines in the printed version of the data set above, but are both actually only one line in the `texts.dat` data file.

12. Upward Bound

Program Name: Upward.java Input File: upward.dat

Given a list of positive integers, you are to find the longest sequence of integers in the list that is strictly increasing.

Input

The first line of input will contain a single integer `n` that indicates the number of lists of positive integers to follow. Each of the next `n` lines will contain fewer than 100 positive integers. Each of integers will be separated by a single space.

Output

For each list, you will print, on a single line, the longest sequence of strictly increasing integers in the list, with each integer followed by a space. If there are two or more strictly increasing sequences within a list that have the same number of elements, you will print the sequence that appears last in the list.

Note: A space is optional at the end of each line.

Example Input File

```
2
7 9 3 1 3 5 7 3 4 6
5 7 3 8 29 34 34 23 12 3 5 7
```

Example Output to Screen

```
1 3 5 7
3 8 29 34
```