# Missing Number

### Problem Statement

Given the sum of four numbers and three of those numbers, find the missing number. All four numbers are positive integers.

### Input Format

- The first line will contain **T**, number of test cases.
- For every test case, the input will contain one integer **S** (the sum of the four numbers), **A, B, C** (three of those four numbers).

### Constraints

1. 0 < **T** <= 100
2. 0 <= **S, A, B, C** <= 2^32

### Output Format

- Print the missing number. Don't forget to print a new line after it.

### Sample Input 0

```
1
10 1 2 3
```

### Sample Output 0

```
4
```

### Sample Input 1

```
3
20 2 4 1
15 5 5 5
30 10 5 3
```

### Sample Output 1

```
13
0
12
```

# Unit Matrix

### Problem Statement

You will be given a matrix of size **N * N**. You need to tell if it is unit matrix or not.

**Note**: A matrix is called a unit matrix if all the primary diagonal elements are 1 and all the non-diagonal elements are 0.

### Input Format

- First line will contain **N**, the row and column of the square matrix.
- Then the N * N sized matrix will be given where the values are integer only.

### Constraints

- 1 <= **N** <= 100
- 0 <= **Values** <= 100

### Output Format

- Ouput "**YES**" if the matrix is unit matrix, otherwise output "**NO**" without the quotation marks.

### Sample Input 0

```
3
1 0 0
0 1 0
0 0 1
```

### Sample Output 0

```
YES
```

### Sample Input 1

```
5
1 0 0 0 0
0 1 0 0 0
0 0 1 0 1
0 0 0 1 0
0 0 0 0 1
```

### Sample Output 1

```
NO
```

# Matrix Again

### Problem Statement

You will be given a 2D matrix of **N * M** size. The matrix will contain integer value only. You need to print the values of last row and then print the values of last column.

### Input Format

- First line will contain **N** and **M** the row and column respectively.
- Then the 2D matrix will be given.

### Constraints

1. 2 <= **N,M** <= 100
2. 0 <= **Element** <= 100

### Output Format

- First line will contain the values of last row.
- Second line will contain the values of last column.

### Sample Input 0

```
3 4
1 2 3 4
5 6 7 8
6 5 4 2
```

### Sample Output 0

```
6 5 4 2
4 8 2
```

### Sample Input 1

```
5 3
1 2 3
5 6 7
6 5 4
3 5 4
1 2 3
```

### Sample Output 1

```
1 2 3
3 7 4 4 3
```

# Count It

### Problem Statement

You will be given a string **S** consisting of small alphabets, capital alphabets and spaces. You need to count number of small alphabets, capital alphabets and spaces in the string S.

### Input Format

- Input will contain a string **S**.

### Constraints

1. 1 <= **|S|** <= 1000; Here |S| means the length of S.

### Output Format

- Output in the format given in the sample output.

### Sample Input 0

```
Hello Everyone
```

### Sample Output 0

```
Capital - 2
Small - 11
Spaces - 1
```

### Sample Input 1

```
I Hope You Have Been Enjoying
```

### Sample Output 1

```
Capital - 6
Small - 18
Spaces - 5
```

# WOW Pattern's Again

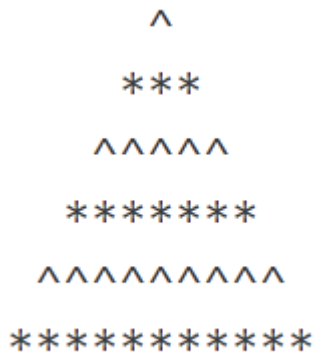
### Problem Statement

Write a C program that will take an integer **N** and creates this pattern for the corresponding inputs.

For example,

- If **N=3**, the pattern look like this,

- If **N=6**, the pattern look like this,



**Input Format**

- You will be given a positive integer **N** as input.

**Constraints**

1. 1 <= **N** <= 20

**Output Format**

- Output the pattern

**Sample Input 0**

```
1
```

**Sample Output 0**

```
^
```

**Sample Input 1**

```
3
```

**Sample Output 1**

```
 ^
```

```
 ***
^^^^^
```

**Sample Input 2**

```
6
```

**Sample Output 2**

```
   ^
  ***
 ^^^^^
 *******
^^^^^^^^^
***********
```

## Who Wins

**Problem Statement**

**Tiger** and **Pathan** are bored with movies and want to play a game. They will play the game in **N** round. Each time, both of them will guess a number **X1** and **X2**, and whoever guesses the larger number will win that round. The final winner will be the one who wins the most rounds. In the event that they both win an equal number of times, the game will end in a draw.

**Input Format**

- First line will contain **N**, the number of rounds.
- Next N lines will contain two numbers **X1** and **X2**, guessed by Tiger and Pathan respectively.

**Constraints**

1. 1 <= **N** <= 1000
2. -10^9 <= **X1, X2** <= 10^9

**Output Format**

- Output "**Tiger**" if Tiger wins, "**Pathan**" if Pathan wins. If it is a draw, print "**Draw**" without the quotation marks.

**Sample Input 0**

```
5
10 20
30 20
15 16
```

```
100 20
50 45
```

### Sample Output 0

```
Tiger
```

### Explanation 0

Tiger won the 2nd, 4th and 5th round. Pathan won the 1st and 3rd round. So, Tiger is the winner!

### Sample Input 1

```
5
10 20
30 20
15 16
20 20
50 45
```

### Sample Output 1

```
Draw
```

### Explanation 1

Tiger won the 2nd and 5th round. Pathan won the 1st and 3rd round. So, it's a draw!

### Sample Input 2

```
4
10 20
20 10
10 30
25 25
```

### Sample Output 2

```
Pathan
```

### Explanation 2

Pathan won the 1st and 3rd round. Tiger won only the 2nd round. So, Pathan is the winner!

## Middle Man

### Problem Statement

There are **N** army people standing in a line. Their major ordered them to make a line in asceding order according to their age. And then the major called the person stading in the

middle. If there are two people in the middle then he will call both of them. Can you tell the age of the middle man or men?

**Note**: If **N** is odd then the middle person will be **(N+1)/2** otherwise there will be two person in the middle, and they are **(N/2)** and **(N/2)+1**.

### Input Format

- First line will contain **N**, the number of army people.
- Second line will contain **N** number which are the ages of those people.

### Constraints

1. 1 <= **N** <= 100
2. 0 <= **Age** <= 1000

### Output Format

- Output the age or ages of the middle man or men.

### Sample Input 0

```
6
20 30 10 50 40 60
```

### Sample Output 0

```
30 40
```

### Explanation 0

After sorting in ascending order, the ages will look like -> 10 20 30 40 50 60
Here there are two values in the middle which are 30 and 40.

### Sample Input 1

```
5
40 20 50 10 30
```

### Sample Output 1

```
30
```

### Explanation 1

After sorting in ascending order, the ages will look like -> 10 20 30 40 50.
Here the age in the middle is 30.