

NAME - MUHILAN SROLL NO. -  
241701033DEPT - CSD

## Week 1-01

QUESTION 00 days 0 hours

Question 1

Correct

Marked out of  
3.00

 Flag question

### Objective

This is a simple challenge to help you practice printing to stdout.

We're starting out by printing the most famous computing phrase of all time! In the editor below, use either printf or cout to print the string **Hello, World!** to stdout.

### Input Format

You do not need to read any input in this challenge.

### Output Format

Print **Hello, World!** to stdout.

## Output Format

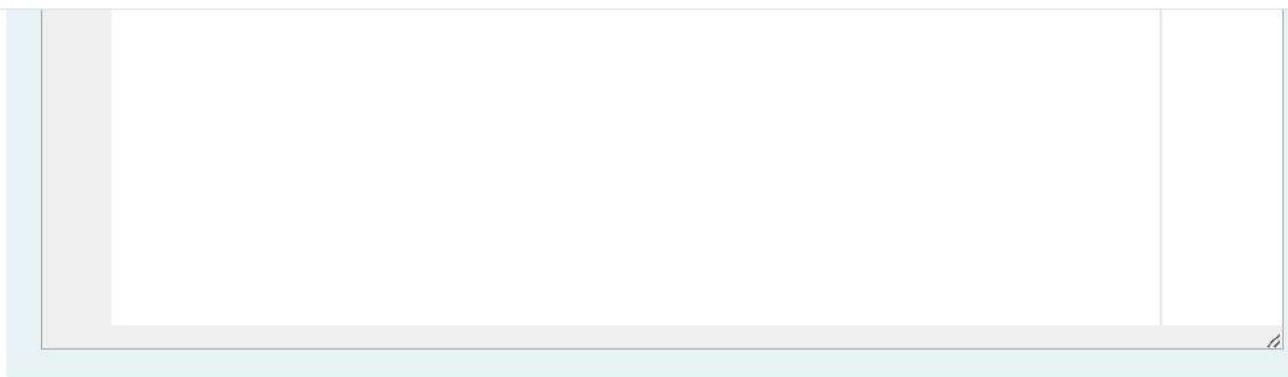
Print **Hello, World!** to stdout.

## Sample Output

Hello, World!

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2
3 int main()
4 {
5     printf("Hello, World!");
6     return 0;
7 }
```



	<b>Expected</b>	<b>Got</b>	
✓	Hello, World!	Hello, World!	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
5.00

Flag question

**Objective**

This challenge will help you to learn how to take a character, a string and a sentence as input in C.

Question **2**

Correct

Marked out of  
5.00

 Flag question

### Objective

This challenge will help you to learn how to take a character, a string and a sentence as input in C.

To take a single character **ch** as input, you can use `scanf("%c", &ch);` and `printf("%c", ch);` writes a character specified by the argument char to stdout:

```
char ch;  
scanf("%c", &ch);  
printf("%c", ch);
```

This piece of code prints the character **ch**.

### Task

You have to print the character, **ch**.

### **Input Format**

Take a character, **ch** as input.

### **Output Format**

Print the character, **ch**.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2
3 int main()
4 {
5     char ch;
6     scanf("%c",&ch);
7     printf("%c",ch);
8     return 0;
9 }
```

```
3 int main()
4 {
5     char ch;
6     scanf("%c",&ch);
7     printf("%c",ch);
8     return 0;
9 }
```

	Input	Expected	Got	
✓	c	c	c	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00[Flag question](#)**Objective**

The fundamental data types in c are int, float and char. Today, we're discussing int and float data types.

The printf() function prints the given statement to the console. The syntax is printf("format string",argument\_list);. In the function, if we are using an integer, character, string or float as argument, then in the format string we have to write %d (integer), %c (character), %s (string), %f (float) respectively.

The scanf() function reads the input data from the console. The syntax is scanf("format string",argument\_list);. For ex: The scanf("%d",&number) statement reads integer number from the console and stores the given value in variable **number**.

To input two integers separated by a space on a single line, the command is scanf("%d %d", &n, &m), where **n** and **m** are the two integers.

**Task**

Your task is to take two numbers of [int data type](#), two numbers of float data type as input and output their sum:

## Task

Your task is to take two numbers of [int data type](#), two numbers of [float data type](#) as input and output their sum:

1. Declare **4** variables: two of type int and two of type float.
2. Read **2** lines of input from stdin (according to the sequence given in the 'Input Format' section below) and initialize your **4** variables.
3. Use the + and - operator to perform the following operations:
  - o Print the sum and difference of two int variable on a new line.
  - o Print the sum and difference of two float variable rounded to one decimal place on a new line.

## Input Format

The first line contains two integers.

The second line contains two floating point numbers.

## Constraints

### **Constraints**

- $1 \leq \text{integer variables} \leq 10^4$
- $1 \leq \text{float variables} \leq 10^4$

### **Output Format**

Print the sum and difference of both integers separated by a space on the first line, and the sum and difference of both float (scaled to **1** decimal place) separated by a space on the second line.

### **Sample Input**

```
10 4  
4.0 2.0
```

### **Sample Output**

### Sample Output

14 6

6.0 2.0

### Explanation

When we sum the integers **10** and **4**, we get the integer **14**. When we subtract the second number **4** from the first number **10**, we get **6** as their difference.

When we sum the floating-point numbers **4.0** and **2.0**, we get **6.0**. When we subtract the second number **2.0** from the first number **4.0**, we get **2.0** as their difference.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,x,y;
5     float c,d,u,v;
6     scanf("%d %d",&a,&b);
7     scanf("%f %f",&c,&d);
```

When we sum the floating-point numbers **4.0** and **2.0**, we get **6.0**. When we subtract the second number **2.0** from the first number **4.0**, we get **2.0** as their difference.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,x,y;
5     float c,d,u,v;
6     scanf("%d %d",&a,&b);
7     scanf("%f %f",&c,&d);
8     x=a+b;
9     y=a-b;
10    u=c+d;
11    v=c-d;
12    printf("%d %d",x,y);
13    printf("\n%.1f %.1f",u,v);
14 }
```

```
13     printf("\n%.1f %.1f",u,v);  
14 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	10 4 4.0 2.0	14 6 6.0 2.0	14 6 6.0 2.0	✓
✓	20 8 8.0 4.0	28 12 12.0 4.0	28 12 12.0 4.0	✓

Passed all tests! ✓

[Finish review](#)

## Week 1 - 02

Question **1**

Correct

Marked out of  
3.00

 Flag question

Write a program to input a name (as a single character) and marks of three tests as m1, m2, and m3 of a student considering all the three marks have been given in integer format.

Now, you need to calculate the average of the given marks and print it along with the name as mentioned in the output format section.

All the test marks are in integers and hence calculate the average in integer as well. That is, you need to print the integer part of the average only and neglect the decimal part.

Input format :

Line 1 : Name(Single character)

Line 2 : Marks scored in the 3 tests separated by single space.

**Input format :**

Line 1 : Name(Single character)

Line 2 : Marks scored in the 3 tests separated by single space.

**Output format :**

First line of output prints the name of the student.

Second line of the output prints the average mark.

**Constraints**

Marks for each student lie in the range 0 to 100 (both inclusive)

**Sample Input 1 :**

A

3 4 6

A  
3 4 6

Sample Output 1 :

A  
4

Sample Input 2 :

T  
7 3 8

Sample Output 2 :

T  
6

Sample Output 2 :

T

6

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     char x;
5     int m1,m2,m3,a;
6     scanf("%c",&x);
7     scanf("%d %d %d",&m1,&m2,&m3);
8     a=(m1+m2+m3)/3;
9     printf("%c",x);
10    printf("\n%d",a);
11 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	A 3 4 6	A 4	A 4	✓
✓	T 7 3 8	T 6	T 6	✓
✓	R 0 100 99	R 66	R 66	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
5.00[Flag question](#)

Some C data types, their format specifiers, and their most common bit widths are as follows:

- *Int* ("%d"): 32 Bit integer
- *Long* ("%ld"): 64 bit integer
- *Char* ("%c"): Character type
- *Float* ("%f"): 32 bit real value
- *Double* ("%lf"): 64 bit real value

**Reading**

To read a data type, use the following syntax:

```
scanf(`formatSpecifier`, &val)
```

For example, to read a *character* followed by a *double*:

```
char ch;  
double d;  
scanf("%c %lf", &ch, &d);
```

For the moment, we can ignore the spacing between format specifiers.

**Printing**

To print a data type, use the following syntax:

## **Printing**

To print a data type, use the following syntax:

```
printf(`formatSpecifier`, val)
```

For example, to print a *character* followed by a *double*:

```
char ch = 'd';
double d = 234.432;
printf("%c %lf", ch, d);
```

**Note:** You can also use *cin* and *cout* instead of *scanf* and *printf*, however, if you are taking a million numbers as input and printing a million lines, it is faster to use *scanf* and *printf*.

## **Input Format**

Input consists of the following space-separated values: *int*, *long*, *char*, *float*, and *double*, respectively.

## **Output Format**

Print each element on a new line in the same order it was received as input. Note that the floating point value should be correct up to 3 decimal places and the double to 9 decimal places.

### **Sample Input**

```
3 12345678912345 a 334.23 14049.30493
```

### **Sample Output**

```
3
```

```
3 12345678912345 a 334.23 14049.30493
```

**Sample Output**

```
3  
12345678912345  
a  
334.230  
14049.304930000
```

**Explanation**

Print *int 3*,  
followed by *long 12345678912345*,  
followed by *char a*,  
followed by *float 334.23*,  
followed by *double 14049.30493*.

**Answer:** (penalty regime: 0 %)

followed by *long* **12345678912345**,

followed by *char* **a**,

followed by *float* **334.23**,

followed by *double* **14049.30493**.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a;
5     long l;
6     char c;
7     float f;
8     double d;
9     scanf("%d %ld %c %f %lf",&a,&l,&c,&f,&d);
10    printf("%d \n%ld \n%c \n%.3f \n%.9lf",a,l,c,f,d);
11
12 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 12345678912345 a 334.23 14049.30493	3 12345678912345 a 334.230 14049.304930000	3 12345678912345 a 334.230 14049.304930000	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00

 [Flag question](#)

Write a program to print the [ASCII value](#) and the two adjacent characters of the given character.

Input

E

Output

69

D F

**Answer:** (penalty regime: 0 %)

D F

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     char a;
5     scanf("%c",&a);
6     printf("%d",a);
7     printf("\n%c %c",a-1,a+1);
8 }
```



	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	E D F	69	69 D F	✓

Passed all tests! ✓

## Week 02 - 01

Question 1

Correct

Marked out of  
3.00

 Flag question

Many people think about their height in feet and inches, even in some countries that primarily use the metric system. Write a program that reads a number of feet from the user, followed by a number of inches. Once these values are read, your program should compute and display the equivalent number of centimeters.

Hint:

One foot is 12 inches.

One inch is 2.54 centimeters.

Input Format

First line, read the number of feet.

Second line, read the number of inches.

Output Format

In one line print the height in centimeters.

Note: All of the values should be displayed using two decimal places.

Sample Input 1

5 6

Sample Output 1

167.64

Two values are taken as input and displayed using two decimal places.

Sample Input 1

5 6

Sample Output 1

167.64

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     float ft,in,j,l,a,ct;
5     scanf("%f",&ft);
6     scanf("%f",&in);
7     j=ft*12;
8     l=j*2.54;
9     a=in*2.54;
10    ct=l+a;
11    printf("%.2f",ct);
12 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	5	167.64	167.64	✓
	6			

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
5.00

 [Flag question](#)

Create a program that reads two integers,  $a$  and  $b$ , from the user. Your program should compute and display:

- The sum of  $a$  and  $b$
- The difference when  $b$  is subtracted from  $a$
- The product of  $a$  and  $b$
- The quotient when  $a$  is divided by  $b$
- The remainder when  $a$  is divided by  $b$

**Input Format**

First line, read the first number.

Second line, read the second number.

**Output Format**

First line, print the sum of  $a$  and  $b$

Second line, print the difference when  $b$  is subtracted from  $a$

Third line, print the product of  $a$  and  $b$

Fourth line, print the quotient when  $a$  is divided by  $b$

Fifth line, print the remainder when  $a$  is divided by  $b$

**Sample**

Input 1 100 6

**Sample Output**

106 94 600 16 4

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b;
5     scanf("%d",&a);
6     scanf("%d",&b);
7     printf("%d", (a+b));
8     printf("\n%d", (a-b));
9     printf("\n%d", (a*b));
10    printf("\n%d", (a/b));
11    printf("\n%d", (a%b));
12 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	100	106	106	✓
	6	94	94	
		600	600	
		16	16	
		4	4	

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00

 [Flag question](#)

A bakery sells loaves of bread for \$3.49 each. Day old bread is discounted by 60 percent. Write a program that begins by reading the number of loaves of day old bread being purchased from the user. Then your program should display the regular price for the bread, the discount because it is a day old, and the total price. Each of these amounts should be displayed on its own line with an appropriate label. All of the values should be displayed using two decimal places.

**Input Format**

Read the number of day old loaves.

**Output Format**

First line, print Regular price: price

Second line, print Discount: discount

Third line, print Total: total

Note: All of the values should be displayed using two decimal places.

**Sample Input 1**

10

**Sample Output 1**

Regular price: 34.90

Discount: 20.94

Total: 13.96

**Answer:** (penalty regime: 0 %)

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     float x,y,dis,no,reg;
5     reg=3.49;
6     scanf("%f",&no);
7     x=reg*no;
8     printf("Regular price: %.2f",x);
9     dis=x*0.6;
10    printf("\nDiscount: %.2f",dis);
11    y=x-dis;
12    printf("\nTotal: %.2f",y);
13 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	10	Regular price: 34.90 Discount: 20.94 Total: 13.96	Regular price: 34.90 Discount: 20.94 Total: 13.96	✓

Passed all tests! ✓

## Week 02 - 02

Question 1

Correct

Marked out of  
3.00

 [Flag question](#)

Goki recently had a breakup, so he wants to have some more friends in his life. Goki has N people who he can be friends with, so he decides to choose among them according to their skills set  $Y_i (1 \leq i \leq n)$ . He wants atleast X skills in his friends. Help Goki find his friends.

**INPUT**

First line contains a single integer X - denoting the minimum skill required to be Goki's friend. Next line contains one integer Y - denoting the skill of the person

• \_\_\_\_\_

**OUTPUT**

Print if he can be friend with Goki. 'YES' (without quotes) if he can be friends with Goki else 'NO' (without quotes).

---

**CONSTRAINTS**

$1 \leq N \leq 1000000$

$1 \leq X, Y \leq 1000000$

**SAMPLE INPUT 1**

100 110

**SAMPLE OUTPUT 1**

YES

1<=N<=1000000

1<=X,Y<=1000000

SAMPLE INPUT 1

100 110

SAMPLE OUTPUT 1

YES

SAMPLE INPUT 2

100 90

SAMPLE OUTPUT 2

NO

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     long int a,b;
5     scanf("%ld %ld",&a,&b);
6     if(b>=a)
7     {
8         printf("YES");
9     }
10    else
11    {
12        printf("NO");
```

100 90

SAMPLE OUTPUT 2

NO

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     long int a,b;
5     scanf("%ld %ld",&a,&b);
6     if(b>=a)
7     {
8         printf("YES");
9     }
10    else
11    {
12        printf("NO");
13    }
14 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	100 110	YES	YES	✓
✓	100 90	NO	NO	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
5.00[Flag question](#)

Before the outbreak of corona virus to the world, a meeting happened in a room in Wuhan. A person who attended that meeting had COVID-19 and no one in the room knew about it! So everyone started shaking hands with everyone else in the room as a gesture of respect and after meeting unfortunately everyone got infected! Given the fact that any two persons shake hand exactly once, Can you tell the total count of handshakes happened in that meeting? Say no to shakehands. Regularly wash your hands. Stay Safe.

**Input Format**

Read an integer N, the total number of people attended that meeting.

**Output Format**

Print the number of handshakes.

**Constraints**

$0 < N < 10^6$

**SAMPLE INPUT 1**

1

**SAMPLE OUTPUT**

0

**SAMPLE INPUT 2**

2

**SAMPLE OUTPUT 2**

SAMPLE OUTPUT

0

SAMPLE INPUT 2

2

SAMPLE OUTPUT 2

1

Explanation Case 1: The lonely board member shakes no hands, hence 0. Case 2: There are 2 board members, 1 handshake takes place.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,n;
5     scanf("%d",&a);
6     n=a*(a-1)/2;
7     printf("%d",n);
8 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	1	0	0	✓
✓	2	1	1	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00[Flag question](#)

In our school days, all of us have enjoyed the Games period. Raghav loves to play cricket and is Captain of his team. He always wanted to win all cricket matches. But only one last Games period is left in school now. After that he will pass out from school. So, this match is very important to him. He does not want to lose it. So he has done a lot of planning to make sure his teams wins. He is worried about only one opponent - Jatin, who is very good batsman. Raghav has figured out 3 types of bowling techniques, that could be most beneficial for dismissing Jatin. He has given points to each of the 3 techniques. You need to tell him which is the maximum point value, so that Raghav can select best technique. 3 numbers are given in input. Output the maximum of these numbers.

**Input:**

Three space separated integers.

**Output:**

Maximum integer value

**SAMPLE INPUT**

8 6 1

**SAMPLE OUTPUT**

8

Explanation Out of given numbers, 8 is maximum.

**Answer:** (penalty regime: 0 %)

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,c;
5     scanf("%d %d %d",&a,&b,&c);
6     if(a>b && a>c)
7     {
8         printf("%d",a);
9     }
10    else if(b>a && b>c)
11    {
12        printf("%d",b);
13    }
14    else
15    {
16        printf("%d",c);
17    }
18 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	81 26 15	81	81	✓

Passed all tests! ✓

## Week 03 - 01

Question 1

Correct

Marked out of  
3.00

[Flag question](#)

Write a program to read two integer values and print true if both the numbers end with the same digit, otherwise print false. Example: If 698 and 768 are given, program should print true as they both end with 8.  
Sample Input 1 25 53 Sample Output 1 false Sample Input 2 27 77 Sample Output 2 true

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,c,d;
5     scanf("%d %d",&a,&b);
6     c=a%10;
7     d=b%10;
8     if(c==d)
9     {
10         printf("true");
11     }
12     else
13     {
14         printf("false");
15     }
16 }
17 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	25 53	false	false	✓
✓	27 77	true	true	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of  
5.00

 Flag question

## Objective

In this challenge, we're getting started with conditional statements.

## Task

Given an integer, ***n***, perform the following conditional actions:

- If ***n*** is odd, print **Weird**
- If ***n*** is even and in the inclusive range of **2** to **5**, print **Not Weird**
- If ***n*** is even and in the inclusive range of **6** to **20**, print **Weird**
- If ***n*** is even and greater than **20**, print **Not Weird**

Complete the stub code provided in your editor to print whether or not ***n*** is weird.

## Input Format

**Input Format**

A single line containing a positive integer, **n**.

**Constraints**

- $1 \leq n \leq 100$

**Output Format**

Print Weird if the number is weird; otherwise, print Not Weird.

**Sample Input 0**

3

**Sample Output 0**

Weird

**Sample Input 1**

24

**Sample Output 1**

Not Weird

**Explanation**

*Sample Case 0: **n = 3***

**n** is odd and odd numbers are weird, so we print **Weird**.

*Sample Case 1: **n = 24***

**n > 20** and **n** is even, so it isn't weird. Thus, we print **Not Weird**.

*Sample Case 1: n = 24*

**n > 20** and **n** is even, so it isn't weird. Thus, we print **Not Weird**.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d",&n);
6     if(n%2!=0)
7     {
8         printf("Weird");
9     }
10    else if(n%2==0 && n>=2 && n<=5)
11    {
12        printf("Not Weird");
13    }
14    else if(n%2==0 && n>=6 && n<=20)
15    {
16        printf("Weird");
17    }
18    else
19    {
20        printf("Not Weird");
21    }
22 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3	Weird	Weird	✓
✓	24	Not Weird	Not Weird	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00[Flag question](#)

Three numbers form a Pythagorean triple if the sum of squares of two numbers is equal to the square of the third. For example, 3, 5 and 4 form a Pythagorean triple, since  $3^2 + 4^2 = 25 = 5^2$ . You are given three integers, a, b, and c. They need not be given in increasing order. If they form a Pythagorean triple, then print "yes", otherwise, print "no". Please note that the output message is in small letters. Sample Input 1 3 5 4  
Sample Output 1 yes Sample Input 2 5 8 2 Sample Output 2 no

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,f,b,c,d,e;
5     scanf("%d %d %d",&a,&b,&c);
6     d=a*a;
7     e=b*b;
8     f=c*c;
9     if(d+e==f)
10    {
11        printf("yes");
12    }
13    else if(f+e==d)
14    {
15        printf("yes");
16    }
17    else if(f+d==e)
18    {
19        printf("yes");
```

```
8     f=c*c;
9     if(d+e==f)
10    {
11        printf("yes");
12    }
13    else if(f+e==d)
14    {
15        printf("yes");
16    }
17    else if(f+d==e)
18    {
19        printf("yes");
20    }
21    else
22    {
23        printf("no");
24    }
25
26 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 5 4	yes	yes	✓
✓	5 8 2	no	no	✓

Passed all tests! ✓

## Week 03 - 02

Question 1

Correct

Marked out of  
3.00

 Flag question

Write a program that determines the name of a shape from its number of sides. Read the number of sides from the user and then report the appropriate name as part of a meaningful message. Your program should support shapes with anywhere from 3 up to (and including) 10 sides. If a number of sides outside of this range is entered then your program should display an appropriate error message.

Sample Input 1

3

Sample Output 1

Triangle

Sample Input 2

7

Sample Input 2

7

Sample Output 2

Heptagon

Sample Input 3

11

Sample Output 3

The number of sides is not supported.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d",&n);
6     switch(n)
7     {
8         case 3:
9             printf("Triangle");
10            break;
11        case 4:
12            printf("Quadrilateral");
13            break;
14        case 5:
15            printf("Pentagon");
16            break;
17        case 6:
18            printf("Hexagon");
19            break;
20        case 7:
21            printf("Heptagon");
22            break;
23        case 8:
24            printf("Octagon");
```

```
9     printf( "triangle" );
10    break;
11    case 4:
12        printf("Quadrilateral");
13        break;
14    case 5:
15        printf("Pentagon");
16        break;
17    case 6:
18        printf("Hexagon");
19        break;
20    case 7:
21        printf("Heptagon");
22        break;
23    case 8:
24        printf("Octagon");
25        break;
26    case 9:
27        printf("Nonagon");
28        break;
29    case 10:
30        printf("Decagon");
31        break;
32    default:
33        printf("The number of sides is not supported.");
34        break;
35    }
36    return 0;
37 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3	Triangle	Triangle	✓
✓	7	Heptagon	Heptagon	✓
✓	11	The number of sides is not supported.	The number of sides is not supported.	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
5.00 Flag question

The Chinese zodiac assigns animals to years in a 12-year cycle. One 12-year cycle is shown in the table below. The pattern repeats from there, with 2012 being another year of the Dragon, and 1999 being another year of the Hare.

Year	Animal
2000	Dragon
2001	Snake
2002	Horse
2003	Sheep
2004	Monkey
2005	Rooster
2006	Dog
2007	Pig
2008	Rat
2009	Ox
2010	Tiger

2006	Dog
2007	Pig
2008	Rat
2009	Ox
2010	Tiger
2011	Hare

Write a program that reads a year from the user and displays the animal associated with that year. Your program should work correctly for any year greater than or equal to zero, not just the ones listed in the table.

**Sample Input 1**

2004

**Sample Output 1**

Monkey

**Sample Input 2**

Sample Output 1

Monkey

Sample Input 2

2010

Sample Output 2

Tiger

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a;
5     scanf("%d",&a);
6     int offset=(a-2000)%12;
7     if(offset<0)
8     {
9         offset+=12;
10    }
11    switch(offset)
12    {
13        case 0:
14            printf("Dragon");
15            break;
16        case 1:
17            printf("Snake");
18            break;
19        case 2:
20            printf("Horse");
21            break;
22        case 3:
23            printf("Sheep");
24            break;
25        case 4:
26            printf("Monkey");
27            break;
28        case 5:
29            printf("Rooster");
```

```
19     case 2:
20     printf("Horse");
21     break;
22     case 3:
23     printf("Sheep");
24     break;
25     case 4:
26     printf("Monkey");
27     break;
28     case 5:
29     printf("Rooster");
30     break;
31     case 6:
32     printf("Dog");
33     break;
34     case 7:
35     printf("Pig");
36     break;
37     case 8:
38     printf("Rat");
39     break;
40     case 9:
41     printf("Ox");
42     break;
43     case 10:
44     printf("Tiger");
45     break;
46     case 11:
47     printf("Hare");
48     break;
49
```

```
34     case 7:
35         printf("Pig");
36         break;
37     case 8:
38         printf("Rat");
39         break;
40     case 9:
41         printf("Ox");
42         break;
43     case 10:
44         printf("Tiger");
45         break;
46     case 11:
47         printf("Hare");
48         break;
49     }
50 }
51 }
52 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	2004	Monkey	Monkey	✓
✓	2010	Tiger	Tiger	✓

Passed all tests! ✓

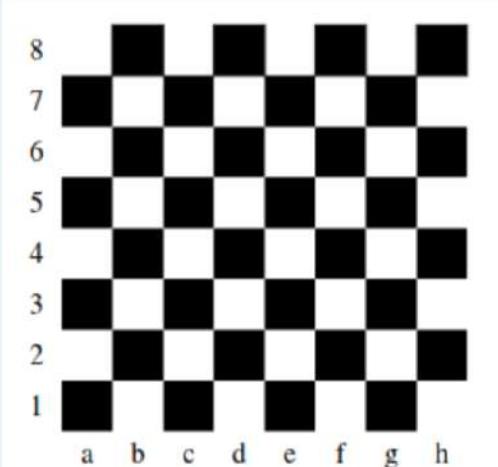
**Question 3**

Correct

Marked out of  
7.00

 Flag question

Positions on a chess board are identified by a letter and a number. The letter identifies the column, while the number identifies the row, as shown below:



Write a program that reads a position from the user. Use an if statement to determine if the column begins with a black square or a white square. Then use modular arithmetic to report the color of the square in that row. For example, if the user enters a1 then your program should report that the square is black. If the user enters d5 then your program should report that the square is white. Your program may assume that a valid position will always be entered. It does not need to perform any error checking.

Sample Input 1

a 1

Sample Output 1

The square is black.

Sample Input 2

d 5

Sample Output 2

The square is white.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     char a;
5     int b;
6     scanf("%c %d",&a,&b);
7     if((a=='a' && b%2!=0) || (a=='c' && b%2!=0) || (a=='e' && b%2!=0) || (a=='g' && b%2!=0))
8     {
9         printf("The square is black.");
10    }
11    else if((a=='b' && b%2==0) || (a=='d' && b%2==0) || (a=='f' && b%2==0) || (a=='h' && b%2==0))
12    {
13        printf("The square is black.");
14    }
15    else if((a=='a' && b%2==0) || (a=='c' && b%2==0) || (a=='e' && b%2==0) || (a=='g' && b%2==0))
16    {
17        printf("The square is white.");
18    }
19    else
20    {
21        printf("The square is white.");
22    }
23 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	a 1	The square is black.	The square is black.	✓
✓	d 5	The square is white.	The square is white.	✓

Passed all tests! ✓

## Week 03 - 03

Question 1

Correct

Marked out of  
3.00

 Flag question

Some data sets specify dates using the year and day of year rather than the year, month, and day of month. The day of year (DOY) is the sequential day number starting with day 1 on January 1st.

There are two calendars - one for normal years with 365 days, and one for leap years with 366 days. Leap years are divisible by 4. Centuries, like 1900, are not leap years unless they are divisible by 400. So, 2000 was a leap year.

To find the day of year number for a standard date, scan down the Jan column to find the day of month, then scan across to the appropriate month column and read the day of year number. Reverse the process to find the standard date for a given day of year.

Write a program to print the Day of Year of a given date, month and year.

Sample Input 1

18  
6  
2020

Sample Input 1

18

6

2020

Sample Output 1

170

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int d,m,y,sum=0;
5     scanf("%d %d %d",&d,&m,&y);
6     int month[100]={31,28,31,30,31,30,31,31,30,31,30,31};
7     for(int i=0;i<m-1;i++)
8     {
9         sum=sum+month[i];
10    }
11    int ntot;
12    int tot=sum+d;
13    if(y%4==0 && y%400==0)
14    {
15        ntot=tot+1;
16        printf("%d",ntot);
17    }
18    else if(y%100!=0)
19    {
20        ntot=tot+1;
21        printf("%d",ntot);
22    }
23    else
24    {
25        printf("%d",tot);
26    }
27 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	18 6 2020	170	170	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of  
5.00

 Flag question

Suppandi is trying to take part in the local village math quiz. In the first round, he is asked about shapes and areas. Suppandi, is confused, he was never any good at math. And also, he is bad at remembering the names of shapes. Instead, you will be helping him [calculate the area](#) of shapes.

- When he says rectangle he is actually referring to a square.
- When he says square, he is actually referring to a triangle.
- When he says triangle he is referring to a rectangle
- And when he is confused, he just says something random. At this point, all you can do is say 0.

Help Suppandi by printing the correct answer in an integer.

**Input Format**

- Name of shape (always in upper case R à Rectangle, S à Square, T à Triangle)
- Length of 1 side
- Length of other side

Note: In case of triangle, you can consider the sides as height and length of base

#### Output Format

- Print the area of the shape.

#### Sample Input 1

```
T  
10  
20
```

#### Sample Output 1

```
200
```

#### Sample Input 2

Sample Output 1

200

Sample Input 2

S

30

40

Sample Output 2

600

Sample Input 3

R  
10  
10

Sample Output 3

100

Sample Input 4

G  
8  
8

Sample Output 4

0

Sample Input

C  
9  
10

Sample Output 4

0

Explanation:

- First is output of area of rectangle
- Then, output of area of triangle
- Then output of area square
- Finally, something random, so we print 0

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,area;
5     char s;
6     scanf("%c %d %d",&s,&a,&b);
7     if(s=='R')
8     {
9         area=a*b;
10    }
11    else if(s=='S')
12    {
13        area=0.5*(a*b);
14    }
15    else if(s=='T')
16    {
17        area=a*b;
18    }
19    else
20    {
21        printf("0");
22    }
23    printf("%d",area);
24 }
25 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	T 10 20	200	200	✓
✓	S 30 40	600	600	✓
✓	B 2 11	0	00	✓
✓	R 10 30	300	300	✓
✓	S 40 50	1000	1000	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00

[Flag question](#)

Superman is planning a journey to his home planet. It is very important for him to know which day he arrives there. They don't follow the 7-day week like us. Instead, they follow a 10-day week with the following days: Day Number Name of Day 1 Sunday 2 Monday 3 Tuesday 4 Wednesday 5 Thursday 6 Friday 7 Saturday 8 Kryptoday 9 Coluday 10 Daxamday Here are the rules of the calendar:

- The calendar starts with Sunday always.
- It has only 296 days. After the 296th day, it goes back to Sunday. You begin your journey on a Sunday and will reach after n. You have to tell on which day you will arrive when you reach there.

Input format:

Contain a number n ( $0 < n$ )

Output format: Print the name of the day you are arriving on

Example Input

7

Example Output

Kryptoday

Example Input

1

Example Output Monday

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int d;
5     scanf("%d",&d);
6     while(d>296)
7     {
8         d=d-296;
9     }
10    while(d>10)
11    {
12        d=d-10;
13    }
14    if(d==1)
15    {
16        printf("Monday");
17    }
18    else if(d==2)
19    {
20        printf("Tuesday");
21    }
22    else if(d==3)
23    {
24        printf("Wednesday");
25    }
26    else if(d==4)
27    {
28        printf("Thursday");
```

```
26     else if(d==4)
27     {
28         printf("Thursday");
29     }
30     else if(d==5)
31     {
32         printf("Friday");
33     }
34     else if(d==6)
35     {
36         printf("Saturday");
37     }
38     else if(d==7)
39     {
40         printf("Kryptonday");
41     }
42     else if(d==8)
43     {
44         printf("Coluday");
45     }
46     else if(d==9)
47     {
48         printf("Daxamday");
49     }
50     else if(d==10)
51     {
52         printf("Sunday");
53     }
54 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	7	Kryptoday	Kryptoday	✓
✓	1	Monday	Monday	✓

Passed all tests! ✓

## Week 04-01

Question **1**

Correct

Marked out of  
3.00

 [Flag question](#)

Alice and Bob are playing a game called "Stone Game". Stone game is a two-player game. Let N be the total number of stones. In each turn, a player can remove either one stone or four stones. The player who picks the last stone, wins. They follow the "Ladies First" norm. Hence Alice is always the one to make the first move. Your task is to find out whether Alice can win, if both play the game optimally.

**Input Format**

First line starts with T, which is the number of test cases. Each test case will contain N number of stones.

**Output Format**

Print "Yes" in the case Alice wins, else print "No".

**Constraints**

$1 \leq T \leq 1000$

$1 \leq T \leq 1000$

$1 \leq N \leq 10000$

Sample Input and Output

Input

3  
1  
6  
7

Output

Yes  
Yes  
No

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int t,i=0,n,T;
5     scanf("%d",&T);
6     while(i<T)
7     {
8         scanf("%d",&n);
9         t=n/4;
10        if(t%2==0 && n%2==0)
11        {
12            printf("No\n");
13        }
14        else if(t%2==1 && n%2==1)
15        {
16
17            printf("No\n");
18        }
19        else
20        {
21            printf("Yes\n");
22        }
23        i++;
24    }
25 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3	Yes	Yes	✓
	1	Yes	Yes	
	6	No	No	
	7			

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
5.00[Flag question](#)

You are designing a poster which prints out numbers with a unique style applied to each of them. The styling is based on the number of closed paths or holes present in a given number.

The number of holes that each of the digits from 0 to 9 have are equal to the number of closed paths in the digit. Their values are:

1, 2, 3, 5, and 7 = 0 holes.

0, 4, 6, and 9 = 1 hole.

8 = 2 holes.

Given a number, you must determine the sum of the number of holes for all of its digits. For example, the number 819 has 3 holes.

Complete the program, it must return an integer denoting the total number of holes in num.

Constraints

$1 \leq \text{num} \leq 109$

$1 \leq \text{num} \leq 10^9$

#### Input Format For Custom Testing

There is one line of text containing a single integer num, the value to process.

#### Sample Input

630

#### Sample Output

2

#### Explanation

Add the holes count for each digit, 6, 3 and 0. Return  $1 + 0 + 1 = 2$ .

#### Sample Case 1

Sample Case 1

Sample Input

1288

Sample Output

4

Explanation

Add the holes count for each digit, 1, 2, 8, 8. Return  $0 + 0 + 2 + 2 = 4$ .

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,sum=0,r;
5     scanf("%d",&a);
6     while(a>0)
7     {
8         r=a%10;
9         if(r==0 || r==9 || r==4 || r==6)
10        {
11            sum=sum+1;
12        }
13        else if(r==8)
14        {
15            sum=sum+2;
16        }
17        a=a/10;
18    }
19    printf("%d",sum);
20 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	630	2	2	✓
✓	1288	4	4	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00[Flag question](#)

The problem solvers have found a new Island for [coding](#) and named it as Philaland. These smart people were given a task to make a purchase of items at the Island easier by distributing various coins with different values. Manish has come up with a solution that if we make coins category starting from \$1 till the maximum price of the item present on Island, then we can purchase any item easily. He added the following example to prove his point.

Let's suppose the maximum price of an item is 5\$ then we can make coins of {\$1, \$2, \$3, \$4, \$5}to purchase any item ranging from \$1 till \$5.

Now Manisha, being a keen observer suggested that we could actually minimize the number of coins required and gave following distribution {\$1, \$2, \$3}. According to him any item can be purchased one time ranging from \$1 to \$5. Everyone was impressed with both of them. Your task is to help Manisha come up with a minimum number of denominations for any arbitrary max price in Philaland.

**Input Format**

Contains an integer N denoting the maximum price of the item present on Philaland.

### **Output Format**

Print a single line denoting the minimum number of denominations of coins required.

### **Constraints**

$1 \leq T \leq 100$

$1 \leq N \leq 5000$

**Refer the sample output for formatting**

### **Sample Input 1:**

10

### **Sample Output 1:**

4

**Sample Output 1:**

4

**Sample Input 2:**

5

**Sample Output 2:**

3

**Explanation:**

For test case 1, N=10.

For test case 1, N=10.

According to Manish  $\{\$1, \$2, \$3, \dots, \$10\}$  must be distributed.

But as per Manisha only  $\{\$1, \$2, \$3, \$4\}$  coins are enough to purchase any item ranging from \$1 to \$10. Hence minimum is 4. Likewise denominations could also be  $\{\$1, \$2, \$3, \$5\}$ . Hence answer is still 4.

For test case 2, N=5.

According to Manish  $\{\$1, \$2, \$3, \$4, \$5\}$  must be distributed.

But as per Manisha only  $\{\$1, \$2, \$3\}$  coins are enough to purchase any item ranging from \$1 to \$5. Hence minimum is 3. Likewise, denominations could also be  $\{\$1, \$2, \$4\}$ . Hence answer is still 3.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n,r=0;
5     scanf("%d",&n);
6     while(n!=0)
7     {
8         n=n/2;
9         r=r+1;
10    }
11    printf("%d",r);
12
13}
14 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	10	4	4	✓
✓	5	3	3	✓
✓	20	5	5	✓
✓	500	9	9	✓
✓	1000	10	10	✓

Passed all tests! ✓

## Week 04 - 02

Question **1**

Correct

Marked out of  
3.00

 [Flag question](#)

A set of N numbers (separated by one space) is passed as input to the program. The program must identify the count of numbers where the number is odd number.

Input Format:

The first line will contain the N numbers separated by one space.

Boundary Conditions:

$3 \leq N \leq 50$

The value of the numbers can be from -99999999 to 99999999

Output Format:

The count of numbers where the numbers are odd numbers.

**Output Format:**

The count of numbers where the numbers are odd numbers.

**Example Input / Output 1:**

**Input:**

5 10 15 20 25 30 35 40 45 50

**Output:**

5

**Explanation:**

The numbers meeting the criteria are 5, 15, 25, 35, 45.

Output:

5

Explanation:

The numbers meeting the criteria are 5, 15, 25, 35, 45.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n,count;
5     char ch;
6     while(scanf("%d",&n)==1)
7     {
8         if(n%2!=0)
9         {
10             count++;
11         }
12         ch=getchar();
13         if(ch=='\n')
14         {
15             break;
16         }
17     }
18     printf("%d",count);
19 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	5 10 15 20 25 30 35 40 45 50	5	5	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of  
5.00

 [Flag question](#)

Given a number N, return true if and only if it is a *confusing number*, which satisfies the following condition:

We can rotate digits by 180 degrees to form new digits. When 0, 1, 6, 8, 9 are rotated 180 degrees, they become 0, 1, 9, 8, 6 respectively. When 2, 3, 4, 5 and 7 are rotated 180 degrees, they become invalid. A *confusing number* is a number that when rotated 180 degrees becomes a **different** number with each digit valid.

**Example 1:**

6 -> 9

Input: 6

Output: true

Explanation:

We get 9 after rotating 6, 9 is a valid number and  $9 \neq 6$ .

**Example 2:**

89 -> 68

Input: 89

**Example 2:**

89 -> 68

Input: 89

Output: true

Explanation:

We get 68 after rotating 89, 86 is a valid number and  $86 \neq 89$ .

**Example 3:**

11 -> 11

Input: 11

Output: false

Explanation:

We get 11 after rotating 11, 11 is a valid number but the value remains the same, thus 11 is not a confusing number.

**Note:**

1.  $0 \leq N \leq 10^9$
2. After the rotation we can ignore leading zeros, for example if after rotation we have 0008 then this

**Explanation:**

We get 11 after rotating 11, 11 is a valid number but the value remains the same, thus 11 is not a confusing number.

**Note:**

1.  $0 \leq N \leq 10^9$
2. After the rotation we can ignore leading zeros, for example if after rotation we have 0008 then this number is considered as just 8.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a, ch;
5     scanf("%d",&a);
6     while(a!=0)
7     {
8         int b=a%10;
9         a=a/10;
10        switch(b)
11        {
12            case 0:
13            case 6:
14            case 8:
15            case 9:
16                ch=0;
17                break;
18            default:
19                ch=1;
20        }
21    }
22    if(ch==1)
23        printf("false");
24    else
25        printf("true");
26
27 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	6	true	true	✓
✓	89	true	true	✓
✓	25	false	false	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00

 [Flag question](#)

A nutritionist is labeling all the best power foods in the market. Every food item arranged in a single line, will have a value beginning from 1 and increasing by 1 for each, until all items have a value associated with them. An item's value is the same as the number of macronutrients it has. For example, food item with value 1 has 1 macronutrient, food item with value 2 has 2 macronutrients, and incrementing in this fashion.

The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients. However, the nutritionist must avoid prescribing a particular sum of macronutrients (an 'unhealthy' number), and this sum is known. The nutritionist chooses food items in the increasing order of their value. Compute the highest total of macronutrients that can be prescribed to a patient, without the sum matching the given 'unhealthy' number.

Here's an illustration:

Given 4 food items (hence value: 1,2,3 and 4), and the unhealthy sum being 6 macronutrients, on choosing items 1, 2, 3 -> the sum is 6, which matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is from among:

- $2 + 3 + 4 = 9$
- $1 + 3 + 4 = 8$

Given 4 food items (hence value: 1,2,3 and 4), and the unhealthy sum being 6 macronutrients, on choosing items 1, 2, 3 -> the sum is 6, which matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is from among:

- $2 + 3 + 4 = 9$
- $1 + 3 + 4 = 8$
- $1 + 2 + 4 = 7$

Since  $2 + 3 + 4 = 9$ , allows for maximum number of macronutrients, 9 is the right answer.

Complete the code in the editor below. It must return an integer that represents the maximum total of macronutrients, modulo  $1000000007$  ( $10^9 + 7$ ).

It has the following:

*n*: an integer that denotes the number of food items

*k*: an integer that denotes the unhealthy number

### Constraints

- $1 \leq n \leq 2 \times 10^9$
- $1 \leq k \leq 4 \times 10^{15}$

$k$ : an integer that denotes the unhealthy number

### Constraints

- $1 \leq n \leq 2 \times 10^9$
- $1 \leq k \leq 4 \times 10^{15}$

Input Format For Custom Testing

The first line contains an integer,  $n$ , that denotes the number of food items.

The second line contains an integer,  $k$ , that denotes the unhealthy number.

### Sample Input 0

```
2
2
```

### Sample Output 0

**Sample Output 0**

3

**Explanation 0**

The following sequence of  $n = 2$  food items:

1. Item 1 has 1 macronutrients.
2.  $1 + 2 = 3$ ; observe that this is the max total, and having avoided having exactly  $k = 2$  macronutrients.

**Sample Input 1**

2

1

**Sample Output 1**

### **Explanation 1**

1. Cannot use item 1 because  $k = 1$  and  $sum \equiv k$  has to be avoided at any time.
2. Hence, max total is achieved by  $sum = 0 + 2 = 2$ .

Sample Case 2

### **Sample Input For Custom Testing**

### **Sample Input 2**

3  
3

### **Sample Output 2**

5

3

**Sample Output 2**

5

**Explanation 2**

$2 + 3 = 5$ , is the best case for maximum nutrients.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     long int n,k,sum;
5     scanf("%ld %ld",&n,&k);
6     sum=0;
7     for(int i=1;i<=n;i++)
8     {
9         sum+=i;
10        if(sum==k)
11            sum-=1;
12    }
13    printf("%ld",sum%1000000007);
14 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	2 2	3	3	✓
✓	2 1	2	2	✓
✓	3 3	5	5	✓

Passed all tests! ✓

## Week 05 - 01

Question **1**

Correct

Marked out of  
3.00

 [Flag question](#)

Write a program that prints a simple chessboard.

Input format:

The first line contains the number of inputs T.

The lines after that contain different values for size of the chessboard

Output format:

Print a chessboard of dimensions size \* size. Print a W for white spaces and B for black spaces.

Input:

2

3

5

2  
3  
5

Output:

WBW  
BWB  
WBW  
WBWBW  
BWBWB  
WBWBW  
BWBWB  
WBWBW

**Answer:** (penalty regime: 0%)

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int T,d,i=0,i1,i2,o;
5     char c;
6     scanf("%d",&T);
7     while(i<T)
8     {
9         scanf("%d",&d);
10        i1=0;
11        while(i1<d)
12        {
13            o=1;
14            i2=0;
15            if(i1%2==0)
16            {
17                o=0;
18            }
19            while(i2<d)
20            {
21                c='B';
22                if(i2%2==o)
23                {
24                    c='W';
25                }
26                printf("%c",c);
```

```
15     if(i1%2==0)
16     {
17         o=0;
18     }
19     while(i2<d)
20     {
21         c='B';
22         if(i2%2==o)
23         {
24             c='W';
25         }
26         printf("%c",c);
27         i2++;
28
29     }
30     i1+=1;
31     printf("\n");
32 }
33     i=i+1;
34 }
35 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	2	WBW	WBW	✓
	3	BWB	BWB	
	5	WBW	WBW	
		WBWBW	WBWBW	
		BWBWB	BWBWB	
		WBWBW	WBWBW	
		BWBWB	BWBWB	
		WBWBW	WBWBW	

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
5.00

 [Flag question](#)

Let's print a chessboard!

Write a program that takes input:

The first line contains T, the number of test cases

Each test case contains an integer N and also the starting character of the chessboard

Output Format

Print the chessboard as per the given examples

Sample Input / Output

Input:

2

2 W

Input:

2  
2 W  
3 B

Output:

WB  
BW  
BWB  
WBW  
BWB

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int T,d,i,i1,i2,o,z;
5     char s,c;
6     scanf("%d",&T);
7     for(i=0;i<T;i++)
8     {
9         scanf("%d %c",&d,&s);
10        for(i1=0;i1<d;i1++)
11        {
12            z=(s=='W') ? 0:1;
13            o=(i1%2==z) ? 0:1;
14            for(i2=0;i2<d;i2++)
15            {
16                c=(i2%2==o) ? 'W':'B';
17                printf("%c",c);
18            }
19            printf("\n");
20        }
21    }
22    return 0;
23 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	2	WB	WB	✓
	2 W	BW	BW	
	3 B	BWB	BWB	
		WBW	WBW	
		BWB	BWB	

Passed all tests! ✓

Question **3**

Correct

Marked out of  
7.00

 [Flag question](#)

Decode the logic and print the Pattern that corresponds to given input.

If N= 3

then pattern will be :

10203010011012

\*\*4050809

\*\*\*\*607

If N= 4, then pattern will be:

1020304017018019020

\*\*50607014015016

\*\*\*\*809012013

\*\*\*\*\*10011

```
**50607014015016
****809012013
*****10011
```

#### Constraints

$2 \leq N \leq 100$

#### Input Format

First line contains  $T$ , the number of test cases

Each test case contains a single integer  $N$

#### Output

First line print Case # $i$  where  $i$  is the test case number

In the subsequent line, print the pattern

#### Test Case 1

Test Case 1

3  
3  
4  
5

Output

Case #1

10203010011012

\*\*4050809

\*\*\*\*607

Case #2

1020304017018019020

\*\*50607014015016

\*\*\*\*809012013

\*\*\*\*\*10011

Case #3

Case #3

102030405026027028029030

\*\*6070809022023024025

\*\*\*\*10011012019020021

\*\*\*\*\*13014017018

\*\*\*\*\*15016

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n,v,p3,c,in,i,i1,i2,t,ti;
5     scanf("%d",&t);
6     for(ti=0;ti<t;ti++)
7     {
8         v=0;
9         scanf("%d",&n);
10        printf("Case #%d\n",ti+1);
11        for(i=0;i<n;i++)
12        {
13            c=0;
14            if(i>0)
15            {
16                for(i1=0;i1<i;i1++) printf("**");
17            }
18            for(i1=i;i1<n;i1++)
19            {
20                if(i>0) c++;
21                printf("%d",c);
22            }
23            if(i==0)
24            {
25                p3=v+(v*(v-1))+1;
26                in=p3;
27            }

```

```
12 v    {
13         c=0;
14         if(i>0)
15     {
16             for(i1=0;i1<i;i1++) printf("**");
17         }
18         for(i1=i;i1<n;i1++)
19     {
20             if(i>0) c++;
21             printf("%d0",++v);
22         }
23         if(i==0)
24     {
25             p3=v+(v*(v-1))+1;
26             in=p3;
27         }
28         in=in-c;
29         p3=in;
30         for(i2=i;i2<n;i2++)
31     {
32             printf("%d",p3++);
33             if(i2!=n-1) printf("0");
34         }printf("\n");
35     }
36 }
37 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 3 4 5	Case #1 10203010011012 **4050809 ****607  Case #2 1020304017018019020 **50607014015016 ****809012013 *****10011  Case #3 102030405026027028029030 **6070809022023024025 ****10011012019020021 *****13014017018 *****15016	Case #1 10203010011012 **4050809 ****607  Case #2 1020304017018019020 **50607014015016 ****809012013 *****10011  Case #3 102030405026027028029030 **6070809022023024025 ****10011012019020021 *****13014017018 *****15016	✓

Passed all tests! ✓

## Week 05 - 02

Question 1

Correct

Marked out of  
3.00

 [Flag question](#)

The k-digit number N is an Armstrong number if and only if the k-th power of each digit sums to N.

Given a positive integer N, return true if and only if it is an Armstrong number.

Example 1:

Input:

153

Output:

true

Explanation:

153 is a 3-digit number, and  $1^3 + 5^3 + 3^3$ .

Example 2:

Input:

123

Output:

false

Explanation:

123 is a 3-digit number, and  $1^3 + 2^3 + 3^3 = 36$ .

Example 3:

Input:

1634

Output:

true

Note:

$1 \leq N \leq 10^8$

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 #include<math.h>
3 int main()
4 {
5     int sum1=0,org,n,no,sum=0,rem1;
6     scanf("%d",&n);
7     no=n;
8     org=n;
9     while(n!=0)
10    {
11         //rem=n%10;
12         //sum=0;
13         sum=sum+1;
14         n=n/10;
15     }
16     while(no!=0)
17    {
18         rem1=no%10;
19         sum1=(pow(rem1,sum))+sum1;
20         no=no/10;
21     }
22     if(org==sum1)
23    {
24         printf("true");
25     }
26     else
27    {
28         printf("false");
29     }
30 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	153	true	true	✓
✓	123	false	false	✓

Passed all tests! ✓

Question 2  
Correct  
Marked out of  
5.00

 Flag question

Take a number, reverse it and add it to the original number until the obtained number is a palindrome.  
Constraints 1<=num<=99999999 Sample Input 1 32 Sample Output 1 55 Sample Input 2 789 Sample Output 2 66066

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     long int org,num,i=0,no,n;
5     scanf("%ld",&num);
6     do
7     {
8         org=num;
9         no=0;
10        while(num!=0)
11        {
12            n=num%10;
13            no=(no*10)+n;
14            num=num/10;
15        }
16        num=org+no;
17        i++;
18    }
19    while(no!=org || i==1);
20    printf("%ld",no);
21 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	32	55	55	✓
✓	789	66066	66066	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of  
7.00

 [Flag question](#)

A number is considered lucky if it contains either 3 or 4 or 3 and 4 both in it. Write a program to print the nth lucky number. Example, 1st lucky number is 3, and 2nd lucky number is 4 and 3rd lucky number is 33 and 4th lucky number is 34 and so on. Note that 13, 40 etc., are not lucky as they have other numbers in it.

The program should accept a number 'n' as input and display the nth lucky number as output.

Sample Input 1:

3

Sample Output 1:

33

Explanation:

Here the lucky numbers are 3, 4, 33, 34, and the 3rd lucky number is 33

Here the lucky numbers are 3, 4, 33, 34., and the 3rd lucky number is 33.

Sample Input 2:

34

Sample Output 2:

33344

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n=1,i=0,nt,co=0,e;
5     scanf("%d",&e);
6     while(i<e)
7     {
8         nt=n;
9         while(nt!=0)
10        {
11            co=0;
12            if(nt%10!=3 && nt%10!=4)
13            {
14                co=1;
15                break;
16            }
17            nt=nt/10;
18        }
19        if(co==0)
20        {
21            i++;
22        }
23        n++;
24    }
25    printf("%d",--n);
26
27 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	34	33344	33344	✓

Passed all tests! ✓

# Week 06 - 01

Question **1**

Correct

Marked out of  
1.00

 Flag question

## Objective

In this challenge, we're going to use loops to help us do some simple math. Check out the Tutorial tab to learn more.

## Task

Given an integer,  $n$ , print its first **10** multiples. Each multiple  $n \times i$  (where  $1 \leq i \leq 10$ ) should be printed on a new line in the form:  $n \times i = \text{result}$ .

## Input Format

A single integer,  $n$ .

## Constraints

**$2 \leq n \leq 20$**

## Output Format

Print **10** lines of output; each line  $i$  (where  $1 \leq i \leq 10$ ) contains the **result** of  $n \times i$  in the form:

$n \times i = \text{result}$ .

## Sample Input

2

## Sample Output

$2 \times 1 = 2$

$n \times i = \text{result.}$

Sample Input

2

Sample Output

$2 \times 1 = 2$

$2 \times 2 = 4$

$2 \times 3 = 6$

$2 \times 4 = 8$

$2 \times 5 = 10$

$2 \times 6 = 12$

$2 \times 7 = 14$

$2 \times 8 = 16$

$2 \times 9 = 18$

$2 \times 10 = 20$

**Answer:** (penalty regime: 0 %)

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d",&n);
6     for(int i=1;i<=10;i++)
7     {
8         printf("%d x %d = %d\n",n,i,n*i);
9     }
10
11 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	2	$2 \times 1 = 2$ $2 \times 2 = 4$ $2 \times 3 = 6$ $2 \times 4 = 8$ $2 \times 5 = 10$ $2 \times 6 = 12$ $2 \times 7 = 14$ $2 \times 8 = 16$ $2 \times 9 = 18$ $2 \times 10 = 20$	$2 \times 1 = 2$ $2 \times 2 = 4$ $2 \times 3 = 6$ $2 \times 4 = 8$ $2 \times 5 = 10$ $2 \times 6 = 12$ $2 \times 7 = 14$ $2 \times 8 = 16$ $2 \times 9 = 18$ $2 \times 10 = 20$	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
1.00[Flag question](#)

A nutritionist is labeling all the best power foods in the market. Every food item arranged in a single line, will have a value beginning from 1 and increasing by 1 for each, until all items have a value associated with them. An item's value is the same as the number of macronutrients it has. For example, food item with value 1 has 1 macronutrient, food item with value 2 has 2 macronutrients, and incrementing in this fashion.

The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients. However, the nutritionist must avoid prescribing a particular sum of macronutrients (an 'unhealthy' number), and this sum is known. The nutritionist chooses food items in the increasing order of their value. Compute the highest total of macronutrients that can be prescribed to a patient, without the sum matching the given 'unhealthy' number.

Here's an illustration:

Given 4 food items (hence value: 1,2,3 and 4), and the unhealthy sum being 6 macronutrients, on choosing items 1, 2, 3 -> the sum is 6, which matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is from among:

- $2 + 3 + 4 = 9$
- $1 + 3 + 4 = 8$
- $1 + 2 + 4 = 7$

Since  $2 + 3 + 4 = 9$ , allows for maximum number of macronutrients, 9 is the right answer.

Complete the code in the editor below. It must return an integer that represents the maximum total of macronutrients, modulo  $1000000007$  ( $10^9 + 7$ ).

It has the following:

- $n$ : an integer that denotes the number of food items
- $k$ : an integer that denotes the unhealthy number

#### Constraints

- $1 \leq n \leq 2 \times 10^9$
- $1 \leq k \leq 4 \times 10^{15}$

#### Input Format For Custom Testing

The first line contains an integer,  $n$ , that denotes the number of food items.

The second line contains an integer,  $k$ , that denotes the unhealthy number.

**Sample Input 0**

2

2

**Sample Output 0**

3

**Explanation 0**

The following sequence of  $n = 2$  food items:

1. Item 1 has 1 macronutrients.
2.  $1 + 2 = 3$ ; observe that this is the max total, and having avoided having exactly  $k = 2$  macronutrients.

**Sample Input 1**

2.  $1 + 2 = 3$ ; observe that this is the max total, and having avoided having exactly  $k = 2$  macronutrients.

**Sample Input 1**

2  
1

**Sample Output 1**

2

**Explanation 1**

1. Cannot use item 1 because  $k = 1$  and  $sum \equiv k$  has to be avoided at any time.
2. Hence, max total is achieved by  $sum = 0 + 2 = 2$ .

Sample Case 2

**Sample Input For Custom Testing****Sample Input 2**

3

3

**Sample Output 2**

5

**Explanation 2**

$2 + 3 = 5$ , is the best case for maximum nutrients.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     long int n,k,sum;
5     scanf("%ld %ld",&n,&k);
6     sum=0;
7     for(int i=1;i<=n;i++)
8     {
9         sum+=i;
10        if(sum==k)
11            sum-=1;
12    }
13    printf("%ld",sum%1000000007);
14 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	2 2	3	3	✓
✓	2 1	2	2	✓
✓	3 3	5	5	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
1.00[Flag question](#)

Determine all positive integer values that evenly divide into a number, its factors. Return the  $p^{th}$  element of your list, sorted ascending. If there is no  $p^{th}$  element, return 0.

For example, given the number  $n = 20$ , its factors are  $\{1,2,4,5,10,20\}$ . Using **1-based indexing** if  $p = 3$ , return 4. If  $p > 6$ , return 0.

Complete the code in the editor below. The function should return a long integer value of the  $p^{th}$  integer factor of  $n$ .

It has the following:

$n$ : an integer

$p$ : an integer

**Constraints**

- $1 \leq n \leq 10^{15}$
- $1 \leq p \leq 10^9$

- $1 \leq n \leq 10^{15}$
- $1 \leq p \leq 10^9$

#### Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer  $n$ , the number to factor.

The second line contains an integer  $p$ , the 1-based index of the factor to return.

#### Sample Input 0

```
10
3
```

#### Sample Output 0

```
5
```

### **Explanation 0**

Factoring  $n = 10$  we get  $\{1, 2, 5, 10\}$ . We then return the  $p = 3^{rd}$  factor as our answer.

### **Sample Input 1**

10

5

### **Sample Output 1**

0

### **Explanation 1**

Factoring  $n = 10$  we get  $\{1, 2, 5, 10\}$ . There are only 4 factors and  $p = 5$ . We return 0 as our answer.

### **Sample Input 2**

**Sample Input 2**

1  
1

**Sample Output 2**

1

**Explanation 2**

Factoring  $n = 1$  we get  $\{1\}$ . We then return the  $p = 1^{\text{st}}$  factor as our answer.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 #include<math.h>
3 #include<stdlib.h>
4 int compare(const void *a, const void *b){
5     return(*(long long*)a-*(long long*)b);
6 }
7 int main(){
8     long long n,p;
9     scanf("%lld %lld",&n,&p);
10    long long factors[100000];
11    int count=0;
12    for(long long i=1;i*i<=n;i++){
13        if(n%i==0){
14            factors[count++]=i;
15            if(i!=n/i){
16                factors[count++]=n/i;
17            }
18        }
19    }
20    qsort(factors,count,sizeof(long long),compare);
21    if(p>count){
22        printf("0\n");
23    }else{
24        printf("%lld\n",factors[p-1]);
25    }
26    return 0;
27 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	10 3	5	5	✓
✓	10 5	0	0	✓
✓	1 1	1	1	✓

Passed all tests! ✓

# Week 07 - 01

**Duration** 5 days 22 hours

Question **1**

Correct

Marked out of  
3.00

 [Flag question](#)

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that  $A[i] - A[j] = k$ ,  $i \neq j$ .

**Input Format**

1. First line is number of test cases T. Following T lines contain:
2. N, followed by N integers of the array
3. The non-negative integer k

**Output format**

Print 1 if such a pair exists and 0 if it doesn't.

**Example**

#### Output format

Print 1 if such a pair exists and 0 if it doesn't.

#### Example

Input:

```
1
3 1 3 5
4
```

Output:

```
1
```

Input:

```
1
```

3 1 3 5

4

Output:

1

Input:

1

3 1 3 5

99

Output:

0

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int t;
5     scanf("%d",&t);
6     while(t--)
7     {
8         int n;
9         scanf("%d",&n);
10        int a[n];
11        for(int i=0;i<n;i++)
12        {
13            scanf("%d",&a[i]);
14        }
15        int k,flag=0;
16        scanf("%d",&k);
17        for(int i=0;i<n;i++)
18        {
19            for(int j=i+1;j<n;j++)
20            {
21                if(a[i]-a[j]==k || a[j]-a[i]==k){flag=1;break;}
22            }
23            if(flag) break;
24        printf("%d\n",flag);
25    }
26 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	1 3 1 3 5 4	1	1	✓
✓	1 3 1 3 5 99	0	0	✓

Passed all tests! ✓

**Question 2**  
Correct  
Marked out of  
5.00

 [Flag question](#)

Sam loves chocolates and starts buying them on the 1st day of the year. Each day of the year,  $x$ , is numbered from 1 to  $Y$ . On days when  $x$  is odd, Sam will buy  $x$  chocolates; on days when  $x$  is even, Sam will not purchase any chocolates.

Complete the code in the editor so that for each day  $N_i$  (where  $1 \leq x \leq N \leq Y$ ) in array  $arr$ , the number of chocolates Sam purchased (during days 1 through  $N$ ) is printed on a new line. This is a function-only challenge, so input is handled for you by the locked stub code in the editor.

#### Input Format

The program takes an array of integers as a parameter.

The locked code in the editor handles reading the following input from `stdin`, assembling it into an array of integers ( $arr$ ), and calling `calculate(arr)`.

The first line of input contains an integer,  $T$  (the number of test cases). Each line  $i$  of the  $T$  subsequent lines describes the  $i$ th test case as an integer,  $N_i$  (the number of days).

#### Constraints

#### Constraints

$1 \leq T \leq 2 \times 10^5$

$1 \leq N \leq 2 \times 10^6$

$1 \leq x \leq N \leq Y$

#### Output Format

For each test case,  $T_i$  in arr, your calculate method should print the total number of chocolates Sam purchased by day  $N_i$  on a new line.

#### Sample Input 0

3

1

2

3

### Sample Output 0

```
1
1
4
```

### Explanation

#### Test Case 0: N = 1

Sam buys 1 chocolate on day 1, giving us a total of 1 chocolate. Thus, we print 1 on a new line.

#### Test Case 1: N = 2

Sam buys 1 chocolate on day 1 and 0 on day 2. This gives us a total of 1 chocolate. Thus, we print 1 on a new line.

#### Test Case 2: N = 3

Sam buys 1 chocolate on day 1, 0 on day 2, and 3 on day 3. This gives us a total of 4 chocolates. Thus, we print

Sam buys 1 chocolate on day 1, giving us a total of 1 chocolate. Thus, we print 1 on a new line.

Test Case 1: N = 2

Sam buys 1 chocolate on day 1 and 0 on day 2. This gives us a total of 1 chocolate. Thus, we print 1 on a new line.

Test Case 2: N = 3

Sam buys 1 chocolate on day 1, 0 on day 2, and 3 on day 3. This gives us a total of 4 chocolates. Thus, we print 4 on a new line.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int t;
5     scanf("%d",&t);
6     while(t--)
7     {
8         int n,c=0;
9         scanf("%d",&n);
10        for(int i=0;i<=n;i++)
11        {
12            if(i%2!=0) c=c+i;
13        }printf("%d\n",c);
14    }
15 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3	1	1	✓
	1	1	1	
	2	4	4	
	3			
✓	10	1296	1296	✓
	71	2500	2500	
	100	1849	1849	
	86	729	729	
	54	400	400	
	40	25	25	
	9	1521	1521	
	77	25	25	
	9	49	49	
	13	2401	2401	
	98			

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
7.00

 [Flag question](#)

The number of goals achieved by two football teams in matches in a league is given in the form of two lists.  
Consider:

- Football team A, has played three matches, and has scored { 1 , 2 , 3 } goals in each match respectively.
- Football team B, has played two matches, and has scored { 2 , 4 } goals in each match respectively.
- Your task is to compute, for each match of team B, the total number of matches of team A, where team A has scored less than or equal to the number of goals scored by team B in that match.
- In the above case:
  - For 2 goals scored by team B in its first match, team A has 2 matches with scores 1 and 2.
  - For 4 goals scored by team B in its second match, team A has 3 matches with scores 1, 2 and 3.

Hence, the answer: {2, 3}.

Complete the code in the editor below. The program must return an array of m positive integers, one for each `maxes[i]` representing the total number of elements `nums[j]` satisfying `nums[j] ≤ maxes[i]` where  $0 \leq j < n$  and  $0 \leq i < m$ , in the given order.

It has the following:

`maxes[i]` representing the total number of elements `nums[j]` satisfying `nums[j] ≤ maxes[i]` where  $0 ≤ j < n$  and  $0 ≤ i < m$ , in the given order.

It has the following:

`nums[nums[0],...nums[n-1]]`: first array of positive integers  
`maxes[maxes[0],...maxes[n-1]]`: second array of positive integers

#### Constraints

- $2 ≤ n, m ≤ 105$
- $1 ≤ \text{nums}[j] ≤ 109$ , where  $0 ≤ j < n$ .
- $1 ≤ \text{maxes}[i] ≤ 109$ , where  $0 ≤ i < m$ .

#### Input Format For Custom Testing

Input from `stdin` will be processed as follows and passed to the function.

The first line contains an integer `n`, the number of elements in `nums`.

The next `n` lines each contain an integer describing `nums[j]` where  $0 ≤ j < n$ .

The first line contains an integer  $n$ , the number of elements in  $\text{nums}$ .

The next  $n$  lines each contain an integer describing  $\text{nums}[j]$  where  $0 \leq j < n$ .

The next line contains an integer  $m$ , the number of elements in  $\text{maxes}$ .

The next  $m$  lines each contain an integer describing  $\text{maxes}[i]$  where  $0 \leq i < m$ .

#### Sample Case 0

#### Sample Input 0

```
4
1
4
2
4
2
3
5
```

5

#### Sample Output 0

2

4

#### Explanation 0

We are given  $n = 4$ ,  $\text{nums} = [1, 4, 2, 4]$ ,  $m = 2$ , and  $\text{maxes} = [3, 5]$ .

1. For  $\text{maxes}[0] = 3$ , we have 2 elements in  $\text{nums}$  ( $\text{nums}[0] = 1$  and  $\text{nums}[2] = 2$ ) that are  $\leq \text{maxes}[0]$ .
2. For  $\text{maxes}[1] = 5$ , we have 4 elements in  $\text{nums}$  ( $\text{nums}[0] = 1$ ,  $\text{nums}[1] = 4$ ,  $\text{nums}[2] = 2$ , and  $\text{nums}[3] = 4$ ) that are  $\leq \text{maxes}[1]$ .

Thus, the function returns the array  $[2, 4]$  as the answer.

#### Sample Case 1

#### Sample Input 1

Sample Input 1

```
5
2
10
5
4
8
4
3
1
7
8
```

Sample Output 1

```
1
0
3
```

3

4

#### Explanation 1

We are given,  $n = 5$ ,  $\text{nums} = [2, 10, 5, 4, 8]$ ,  $m = 4$ , and  $\text{maxes} = [3, 1, 7, 8]$ .

1. For  $\text{maxes}[0] = 3$ , we have 1 element in  $\text{nums}$  ( $\text{nums}[0] = 2$ ) that is  $\leq \text{maxes}[0]$ .
2. For  $\text{maxes}[1] = 1$ , there are 0 elements in  $\text{nums}$  that are  $\leq \text{maxes}[1]$ .
3. For  $\text{maxes}[2] = 7$ , we have 3 elements in  $\text{nums}$  ( $\text{nums}[0] = 2$ ,  $\text{nums}[2] = 5$ , and  $\text{nums}[3] = 4$ ) that are  $\leq \text{maxes}[2]$ .
4. For  $\text{maxes}[3] = 8$ , we have 4 elements in  $\text{nums}$  ( $\text{nums}[0] = 2$ ,  $\text{nums}[2] = 5$ ,  $\text{nums}[3] = 4$ , and  $\text{nums}[4] = 8$ ) that are  $\leq \text{maxes}[3]$ .

Thus, the function returns the array  $[1, 0, 3, 4]$  as the answer.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n,m,na[100],ma[100],ans;
5     scanf("%d",&n);
6     for(int i=0;i<n;i++)
7     {
8         scanf("%d",&na[i]);
9     }
10    scanf("%d",&m);
11    for(int i=0;i<n;i++)
12    {
13        scanf("%d",&ma[i]);
14    }
15    for(int j=0;j<m;j++)
16    {
17        ans=0;
18        for(int i=0;i<n;i++)
19        {
20            if(ma[j]>=na[i])
21                ans++;
22        }printf("%d\n",ans);
23    }
24 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	4 1 4 2 4 2 3 5	2 4	2 4	✓
✓	5 2 10 5 4 8 4 3 1 7 8	1 0 3 4	1 0 3 4	✓

Passed all tests! ✓

## Week 06 - 02

QUESTION    2 days 21 hours

Question **1**

Correct

Marked out of  
5.00

 [Flag question](#)

Sunny and Johnny like to pool their money and go to the ice cream parlor. Johnny never buys the same flavor that Sunny does. The only other rule they have is that they spend all of their money.

Given a list of prices for the flavors of ice cream, select the two that will cost all of the money they have.

For example, they have  $m = 6$  to spend and there are flavors costing  $\text{cost} = [1, 2, 3, 4, 5, 6]$ . The two flavors costing **1** and **5** meet the criteria. Using **1**-based indexing, they are at indices **1** and **4**.

### Function Description

Complete the code in the editor below. It should return an array containing the indices of the prices of the two flavors they buy.

It has the following:

- $m$ : an integer denoting the amount of money they have to spend
- $\text{cost}$ : an integer array denoting the cost of each flavor of ice cream

### Input Format

The first line contains an integer,  $t$ , denoting the number of trips to the ice cream parlor. The next  $t$  sets of lines each describe a visit. Each trip is described as follows:

1. The integer  $m$ , the amount of money they have pooled.
2. The integer  $n$ , the number of flavors offered at the time.
3.  $n$  space-separated integers denoting the cost of each flavor:  $\text{cost}[\text{cost}[1], \text{cost}[2], \dots, \text{cost}[n]]$ .

**Note:** The index within the cost array represents the flavor of the ice cream purchased.

### Constraints

- $1 \leq t \leq 50$
- $2 \leq m \leq 10^4$
- $2 \leq n \leq 10^4$
- $1 \leq \text{cost}[i] \leq 10^4$ , " $i \in [1, n]$ "
- There will always be a unique solution.

- There will always be a unique solution.

### **Output Format**

For each test case, print two space-separated integers denoting the indices of the two flavors purchased, in ascending order.

### **Sample Input**

```
2
4
5
1 4 5 3 2
4
4
2 2 4 3
```

### **Sample Output**

1 4

1 2

### **Explanation**

Sunny and Johnny make the following two trips to the parlor:

1. The first time, they pool together  $m = 4$  dollars. Of the five flavors available that day, flavors **1** and **4** have a total cost of **1 + 3 = 4**.
2. The second time, they pool together  $m = 4$  dollars. Of the four flavors available that day, flavors **1** and **2** have a total cost of **2 + 2 = 4**.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int t;
5     scanf("%d",&t);
6     while(t--)
7     {
8         int m,n,i,j,cost[100],a,b;
9         scanf("%d",&m);
10        scanf("%d",&n);
11        for(int i1=0;i1<n;i1++)
12        {
13            scanf("%d",&cost[i1]);
14        }
15        for(j=0;j<n;j++)
16        {
17            for(i=j+1;i<n;i++)
18            {
19                if(cost[i]+cost[j]==m)
20                {
21                    a=j;
22                    b=i;
23                    break;
24                }
25            }
26        }
}
```

```
8     int m,n,i,j,cost[100],a,b;
9     scanf("%d",&m);
10    scanf("%d",&n);
11    for(int i1=0;i1<n;i1++)
12    {
13        scanf("%d",&cost[i1]);
14    }
15    for(j=0;j<n;j++)
16    {
17        for(i=j+1;i<n;i++)
18        {
19            if(cost[i]+cost[j]==m)
20            {
21                a=j;
22                b=i;
23                break;
24            }
25        }
26    }
27    printf("%d %d\n",a+1,b+1);
28 }
29 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	2 4 5 1 4 5 3 2 4 4 2 2 4 3	1 4 1 2	1 4 1 2	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
5.00[Flag question](#)

Numeros the Artist had two lists that were permutations of one another. He was very proud. Unfortunately, while transporting them from one exhibition to another, some numbers were lost out of the first list. Can you find the missing numbers?

As an example, the array with some numbers missing, **arr** = [7, 2, 5, 3, 5, 3]. The original array of numbers **brr** = [7, 2, 5, 4, 6, 3, 5, 3]. The numbers missing are [4, 6].

**Notes**

- If a number occurs multiple times in the lists, you must ensure that the frequency of that number in both lists is the same. If that is not the case, then it is also a missing number.
- You have to print all the missing numbers in ascending order.
- Print each missing number once, even if it is missing multiple times.
- The difference between maximum and minimum number in the second list is less than or equal to **100**.

Complete the code in the editor below. It should return an array of missing numbers.

It has the following:

It has the following:

- arr: the array with missing numbers
- brr: the original array of numbers

### Input Format

There will be four lines of input:

**n** - the size of the first list, **arr**

The next line contains **n** space-separated integers **arr[i]**

**m** - the size of the second list, **brr**

The next line contains **m** space-separated integers **brr[i]**

### Constraints

- $1 \leq n, m \leq 2 \times 10^5$
- $n \leq m$
- $1 \leq brr[i] \leq 2 \times 10^4$
- $X_{max} - X_{min} < 101$

**Output Format**

Output the missing numbers in ascending order.

**Sample Input**

```
10
203 204 205 206 207 208 203 204 205 206
13
203 204 204 205 206 206 207 205 208 203 206 205 206 204
```

**Sample Output**

```
204 205 206
```

**Explanation**

204 205 206

### Explanation

**204** is present in both arrays. Its frequency in ***arr*** is **2**, while its frequency in ***brr*** is **3**. Similarly, **205** and **206** occur twice in ***arr***, but three times in ***brr***. The rest of the numbers have the same frequencies in both lists.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n,m,c,c1=0,co;
5     scanf("%d",&n);
6     int arr[n];
7     for(int a=0;a<n;a++)
8     {
9         scanf("%d",&arr[a]);
10    }
11    scanf("%d",&m);
12    int brr[m],ans[m];
13    for(int b=0;b<m;b++)
14    {
15        scanf("%d",&brr[b]);
16    }
17    for(int j=0;j<m;j++)
18    {
19        c=0;
20        for(int i=0;i<n;i++)
21        {
22            if(arr[i]==brr[j])
23            {
24                c=1;
25                arr[i]=-1;
26                break;
27            }
28        }
    }
```

```
24         c=1;
25         arr[i]=-1;
26         break;
27     }
28 }
29 if(c==0)
30 {
31     ans[c1]=brr[j];
32     c1++;
33 }
34 }
35 for(int a=0;a<c1;a++)
36 {
37     co=0;
38     for(int b=0;b<c1;b++)
39     {
40         if(ans[b]<ans[a])
41             co++;
42     }
43     int temp=ans[a];
44     ans[a]=ans[co];
45     ans[co]=temp;
46 }
47 for(int i=0;i<c1;i++)
48 printf("%d ",ans[i]);
49 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	10 203 204 205 206 207 208 203 204 205 206 13 203 204 204 205 206 207 205 208 203 206 205 206 204	204 205 206	204 205 206	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
5.00[Flag question](#)

Watson gives Sherlock an array of integers. His challenge is to find an element of the array such that the sum of all elements to the left is equal to the sum of all elements to the right. For instance, given the array `arr = [5, 6, 8, 11]`, **8** is between two subarrays that sum to **11**. If your starting array is **[1]**, that element satisfies the rule as left and right sum to **0**.

You will be given arrays of integers and must determine whether there is an element that meets the criterion.

Complete the code in the editor below. It should return a string, either YES if there is an element meeting the criterion or NO otherwise.

It has the following:

- arr: an array of integers

**Input Format**

The first line contains **T**, the number of test cases.

The next **T** pairs of lines each represent a test case.

- The first line contains **n**, the number of elements in the array `arr`.
- The second line contains **n** space-separated integers `arr[i]` where  $0 \leq i < n$ .

### Constraints

- $1 \leq T \leq 10$
- $1 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 2 \times 10^4$
- $0 \leq i \leq n$

### Output Format

For each test case print YES if there exists an element in the array, such that the sum of the elements on its left is equal to the sum of the elements on its right; otherwise print NO.

### Sample Input 0

```
2
3
1 2 3
```

4  
1 2 3 3

#### **Sample Output 0**

NO  
YES

#### **Explanation 0**

For the first test case, no such index exists.

For the second test case,  $\text{arr}[0] + \text{arr}[1] = \text{arr}[3]$ , therefore index **2** satisfies the given conditions.

#### **Sample Input 1**

3  
5

```
1 1 4 1 1
```

```
4
```

```
2 0 0 0
```

```
4
```

```
0 0 2 0
```

### Sample Output 1

```
YES
```

```
YES
```

```
YES
```

### Explanation 1

In the first test case,  $\text{arr}[2] = 4$  is between two subarrays summing to **2**.

In the second case,  $\text{arr}[0] = 2$  is between two subarrays summing to **0**.

In the third case,  $\text{arr}[2] = 2$  is between two subarrays summing to **0**.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int t,n,Is,rs,m;
5     scanf("%d",&t);
6     for(int i=0;i<t;i++)
7     {
8         Is=0;
9         rs=0;
10        scanf("%d",&n);
11        int arr[n];
12        for(int j=0;j<n;j++)
13            scanf("%d",&arr[j]);
14        m=n/2;
15        if(arr[m]==0){
16            for(m=0;arr[m]==0 && m<n;m++);
17        }
18        for(int j=0;j<=m;j++)
19            Is=Is+arr[j];
20        for(int j=m;j<n;j++)
21            rs=rs+arr[j];
22        printf("%s\n", (Is==rs)? "YES" : "NO");
23    }
24 }
25 }
26 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 5 1 1 4 1 1 4 2 0 0 0 4 0 0 2 0	YES YES YES YES	YES YES YES	✓
✓	2 3 1 2 3 4 1 2 3 3	NO YES	NO YES	✓

Passed all tests! ✓

## Week 07 - 02

Duration 3 days 3 hours

Question 1

Correct

Marked out of  
1.00

 Flag question

Coders here is a simple task for you, you have given an array of size **N** and an integer **M**.

Your task is to calculate the **difference between maximum sum and minimum sum of N-M elements** of the given array.

### Constraints:

**$1 \leq t \leq 10$**

**$1 \leq n \leq 1000$**

**$1 \leq a[i] \leq 1000$**

### Input:

First line contains an integer **T** denoting the number of testcases.

First line of every testcase contains two integer **N** and **M**.

Next line contains **N** space separated integers denoting the elements of array

**Output:**

For every test case print your answer in new line

## SAMPLE INPUT

1  
5 1  
1 2 3 4 5

## SAMPLE OUTPUT

4

## Explanation

M is 1 and N is 5 so you have to calculate maximum and minimum sum using (5-1 =) 4 elements.

## Explanation

M is 1 and N is 5 so you have to calculate maximum and minimum sum using (5-1 =) 4 elements.

Maximum sum using the 4 elements would be (2+3+4+5=)14.

Minimum sum using the 4 elements would be (1+2+3+4=)10.

Difference will be 14-10=4.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int t;
5     scanf("%d",&t);
6     while(t--)
7     {
8         int n,m,d,min,temp;
9         scanf("%d %d",&n,&m);
10        d=n-m;
11        int arr[n];
12        for(int i=0;i<n;i++)
13        {
14            scanf("%d",&arr[i]);
15        }
16        for(int j=0;j<n;j++)
17        {
```

```
13 v
14     scanf("%d",&arr[i]);
15 }
16 for(int j=0;j<n;j++)
17 {
18     min=j;
19     for(int k=j;k<n;k++)
20     {
21         if(arr[k]<arr[min])
22             min=k;
23     }
24     temp=arr[min];
25     arr[min]=arr[j];
26     arr[j]=temp;
27 }
28 int maxsum=0,minsum=0;
29 for(int a=0;a<d;a++)
30     minsum+=arr[a];
31     for(int b=n-1;b>m-1;b--)
32         maxsum+=arr[b];
33     printf("%d\n",maxsum-minsum);
34 }
35 return 0;
36 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	1 5 1 1 2 3 4 5	4	4	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
1.00[Flag question](#)

A new deadly virus has infected large population of a planet. A brilliant scientist has discovered a new strain of virus which can cure this disease. Vaccine produced from this virus has various strength depending on midichlorians count. A person is cured only if midichlorians count in vaccine batch is more than midichlorians count of person. A doctor receives a new set of report which contains midichlorians count of each infected patient, Practo stores all vaccine doctor has and their midichlorians count. You need to determine if doctor can save all patients with the vaccines he has. The number of vaccines and patients are equal.

**Input Format**

First line contains the number of vaccines - N. Second line contains N integers, which are strength of vaccines. Third line contains N integers, which are midichlorians count of patients.

**Output Format**

Print a single line containing '**Yes**' or '**No**'.

**Input Constraint**

**Output Format**

Print a single line containing '**Yes**' or '**No**'.

**Input Constraint**

**$1 < N < 10$**

Strength of vaccines and midichlorians count of patients fit in integer.

**SAMPLE INPUT**

```
5
123 146 454 542 456
100 328 248 689 200
```

**SAMPLE OUTPUT**

```
No
```

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n,min1,min2,temp,flag=1;
5     scanf("%d",&n);
6     int vac[n],pat[n];
7     for(int i=0;i<n;i++)
8         scanf("%d",&vac[i]);
9     for(int j=0;j<n-1;j++)
10    {
11         min1=j,min2=j;
12         for(int k=j;k<n;k++)
13         {
14             if(vac[n]<vac[min1])
15                 min1=k;
16             if(vac[n]<vac[min2])
17                 min2=k;
18         }
19         temp=vac[min1];
20         vac[min1]=vac[j];
21         vac[j]=temp;
22         temp=pat[min2];
23         pat[min2]=pat[j];
24         pat[j]=temp;
25     }
26     for(int i=0;i<n;i++)
27     {
28         if(vac[i]<=pat[i])
29         {
```

```
16     if(vac[n]<vac[min2])
17         min2=k;
18     }
19     temp=vac[min1];
20     vac[min1]=vac[j];
21     vac[j]=temp;
22     temp=pat[min2];
23     pat[min2]=pat[j];
24     pat[j]=temp;
25 }
26 for(int i=0;i<n;i++)
27 {
28     if(vac[i]<=pat[i])
29     {
30         flag=0;
31         break;
32     }
33 }
34 if(flag==1)
35 printf("Yes");
36 else
37 printf("No");
38 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	5 123 146 454 542 456 100 328 248 689 200	No	No	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
1.00[Flag question](#)

You are given an array of  $n$  integer numbers  $a_1, a_2, \dots, a_n$ . Calculate the number of pair of indices  $(i, j)$  such that  $1 \leq i < j \leq n$  and  $a_i \text{ xor } a_j = 0$ .

**Input format**

- First line:  $n$  denoting the number of array elements
- Second line:  $n$  space separated integers  $a_1, a_2, \dots, a_n$ .

**Output format**

Output the required number of pairs.

**Constraints**

$$1 \leq n \leq 10^6$$

$$1 \leq a_i \leq 10^9$$

**SAMPLE INPUT**

**SAMPLE INPUT**

5  
1 3 1 4 3

**SAMPLE OUTPUT**

2

**Explanation**

The 2 pair of indices are **(1, 3)** and **(2,5)**.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n, count=0;
5     scanf("%d", &n);
6     int arr[n];
7     for(int i=0; i<n; i++)
8     {
9         scanf("%d", &arr[i]);
10    }
11    for(int i=0; i<n; i++)
12    {
13        for(int j=i+1; j<n; j++)
14        {
15            if((arr[i]^arr[j])==0)
16                count++;
17        }
18    }
19    printf("%d", count);
20 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	5 1 3 1 4 3	2	2	✓

Passed all tests! ✓

Question **4**

Correct

Marked out of  
1.00

 Flag question

You are given an array **A** of non-negative integers of size **m**. Your task is to sort the array in non-decreasing order and print out the original indices of the new sorted array.

**Example:**

$A=\{4,5,3,7,1\}$

After sorting the new array becomes  $A=\{1,3,4,5,7\}$ .

The required output should be "4 2 0 1 3"

**INPUT :**

The first line of input consists of the size of the array

The next line consists of the array of size m

**OUTPUT :**

**OUTPUT:**

Output consists of a single line of integers

**CONSTRAINTS:**

**$1 \leq m \leq 106$**

**$0 \leq A[i] \leq 106$**

NOTE: The indexing of the array starts with 0.

**SAMPLE INPUT**

5

4 5 3 7 1

**SAMPLE OUTPUT**

4 2 0 1 3

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main(){
3     int n;
4     scanf("%d",&n);
5     int arr[n];
6     for(int i=0;i<n;i++)
7         scanf("%d",&arr[i]);
8     int max=arr[0];
9     for(int i=1;i<n;i++)
10    {
11         if(arr[i]>max)
12             max=arr[i];
13     }
14     max++;
15     int min=0;
16     for(int a=0;a<n;a++)
17    {
18         for(int b=0;b<n;b++)
19        {
20             if(arr[b]<arr[min])
21                 min=b;
22         }
23         printf("%d ",min);
24         arr[min]=max;
25     }
26 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	5 4 5 3 7 1	4 2 0 1 3	4 2 0 1 3	✓

Passed all tests! ✓

## Week 09 - 01

Question 1

Correct

Marked out of  
1.00

 [Flag question](#)

You are given a two-dimensional 3\*3 array starting from A [0][0]. You should add the alternate elements of the array and print its sum. It should print two different numbers the first being sum of A 0 0, A 0 2, A 1 1, A 2 0, A 2 2 and A 0 1, A 1 0, A 1 2, A 2 1.

### Input Format

First and only line contains the value of array separated by single space.

A 0 0	A 0 1	A 0 2
4	6	9
A 1 0	A 1 1	A 1 2
2	5	8
A 2 0	A 2 1	A 2 2
1	3	7

1	3	7
---	---	---

### **Output Format**

First line should print sum of A 0 0, A 0 2, A 1 1, A 2 0, A 2 2

Second line should print sum of A 0 1, A 1 0, A 1 2, A 2 1

### **SAMPLE INPUT**

1 2 3 4 5 6 7 8 9

### **SAMPLE OUTPUT**

25

20

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int arr[3][3];
5     for(int i=0;i<3;i++)
6     {
7         for(int j=0;j<3;j++)
8         {
9             scanf("%d",&arr[i][j]);
10        }
11    }
12    int odd=0,even=0;
13    for(int i=0;i<3;i++)
14    {
15        for(int j=0;j<3;j++)
16        {
17            if((i+j)%2!=0)
18                odd+=arr[i][j];
19            else
20                even+=arr[i][j];
21        }
22    }
23 }
24 printf("%d\n%d",even,odd);
25 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	1 2 3 4 5 6 7 8 9	25 20	25 20	✓
✓	21 422 423 443 586 645 657 846 904	2591 2356	2591 2356	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of  
5.00

 [Flag question](#)

Microsoft has come to hire interns from your college. N students got shortlisted out of which few were males and a few females. All the students have been assigned talent levels. Smaller the talent level, lesser is your chance to be selected. Microsoft wants to create the result list where it wants the candidates sorted according to their talent levels, but there is a catch. This time Microsoft wants to hire female candidates first and then male candidates.

The task is to create a list where first all-female candidates are sorted in a descending order and then male candidates are sorted in a descending order.

**Input Format**

The first line contains an integer N denoting the number of students. Next, N lines contain two space-separated integers, ai and bi.

The first integer, ai will be either 1(for a male candidate) or 0(for female candidate).

The second integer, bi will be the candidate's talent level.

The second integer,  $b_i$  will be the candidate's talent level.

#### Constraints

**$1 \leq N \leq 10^5$**

**$0 \leq a_i \leq 1$**

**$1 \leq b_i \leq 10^9$**

#### Output Format

Output space-separated integers, which first contains the talent levels of all female candidates sorted in descending order and then the talent levels of male candidates in descending order.

#### SAMPLE INPUT

5

0 3

1 6

5  
0 3  
1 6  
0 2  
0 7  
1 15

SAMPLE OUTPUT

7 3 2 15 6

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 struct data
3 {
4     int gen,int tal;
5 };
6 int main()
7 {
8     int n;
9     scanf("%d",&n);
10    struct data a[n];
11    for(int i=0;i<n;i++)
12        scanf("%d %d",&a[i].gen,&a[i].tal);
13    for(int i=0;i<n-1;i++)
14    {
15        for(int j=0;j<n-i-1;++j)
16        {
17            if(a[j].tal<a[j+1].tal)
18            {
19                struct data temp=a[j];
20                a[j]=a[j+1];
21                a[j+1]=temp;
22            }
23        }
24    }
25    for(int i=0;i<n;i++)
26    {
27        if(a[i].gen==0)
28            printf("%d ",a[i].tal);
```

```
13
14 {
15     for(int j=0;j<n-i-1;++j)
16     {
17         if(a[j].tal<a[j+1].tal)
18         {
19             struct data temp=a[j];
20             a[j]=a[j+1];
21             a[j+1]=temp;
22         }
23     }
24 }
25 for(int i=0;i<n;i++)
26 {
27     if(a[i].gen==0)
28     printf("%d ",a[i].tal);
29
30 }
31 for(int i=0;i<n;++i)
32 {
33     if(a[i].gen==1)
34     printf("%d ",a[i].tal);
35 }
36 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	5 0 3 1 6 0 2 0 7 1 15	7 3 2 15 6	7 3 2 15 6	✓
✓	6 0 1 0 26 0 39 0 37 0 7 0 13	39 37 26 13 7 1	39 37 26 13 7 1	✓
✓	12 1 12 1 14 1 18 1 1 1 2 1 3 1 5 1 8	31 29 18 14 12 10 9 8 5 3 2 1	31 29 18 14 12 10 9 8 5 3 2 1	✓

	- --	1 14 1 18 1 1 1 2 1 3 1 5 1 8 1 9 1 10 0 29 0 31			
✓	12 0 12 1 12 0 12 1 12 0 12 0 12 1 12 0 12 1 12 1 12 0 12 1 12	12 12	12 12	12 12	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of  
1.00

 [Flag question](#)

Shyam Lal, a wealthy landlord from the state of Rajasthan, being an old fellow and tired of doing hard work, decided to sell all his farmland and to live rest of his life with that money. No other farmer is rich enough to buy all his land so he decided to partition the land into rectangular plots of different sizes with different cost per unit area. So, he sold these plots to the farmers but made a mistake. Being illiterate, he made partitions that could be overlapping. When the farmers came to know about it, they ran to him for compensation of extra money they paid to him. So, he decided to return all the money to the farmers of that land which was overlapping with other farmer's land to settle down the conflict. All the portion of conflicted land will be taken back by the landlord.

To decide the total compensation, he has to calculate the total amount of money to return back to farmers with the same cost they had purchased from him. Suppose, Shyam Lal has a total land area of **1000 x 1000** equal square blocks where each block is equivalent to a unit square area which can be represented on the co-ordinate axis. Now find the total amount of money, he has to return to the farmers. Help Shyam Lal to accomplish this task.

**Input Format:**

The first line of the input contains an integer **N**, denoting the total number of land pieces he had distributed. Next **N** line contains the **5** space separated integers **(X1, Y1), (X2, Y2)** to represent a rectangular piece of land, and cost per unit area **C**.

The first line of the input contains an integer  **$N$** , denoting the total number of land pieces he had distributed. Next  **$N$**  line contains the **5** space separated integers  **$(X_1, Y_1)$ ,  $(X_2, Y_2)$**  to represent a rectangular piece of land, and cost per unit area  **$C$** .

**$(X_1, Y_1)$**  and  **$(X_2, Y_2)$**  are the locations of first and last square block on the diagonal of the rectangular region.

Output Format:

Print the total amount he has to return to farmers to solve the conflict.

Constraints:

**$1 \leq N \leq 100$**

**$1 \leq X_1 \leq X_2 \leq 1000$**

**$1 \leq Y_1 \leq Y_2 \leq 1000$**

**$1 \leq C \leq 1000$**

SAMPLE INPUT

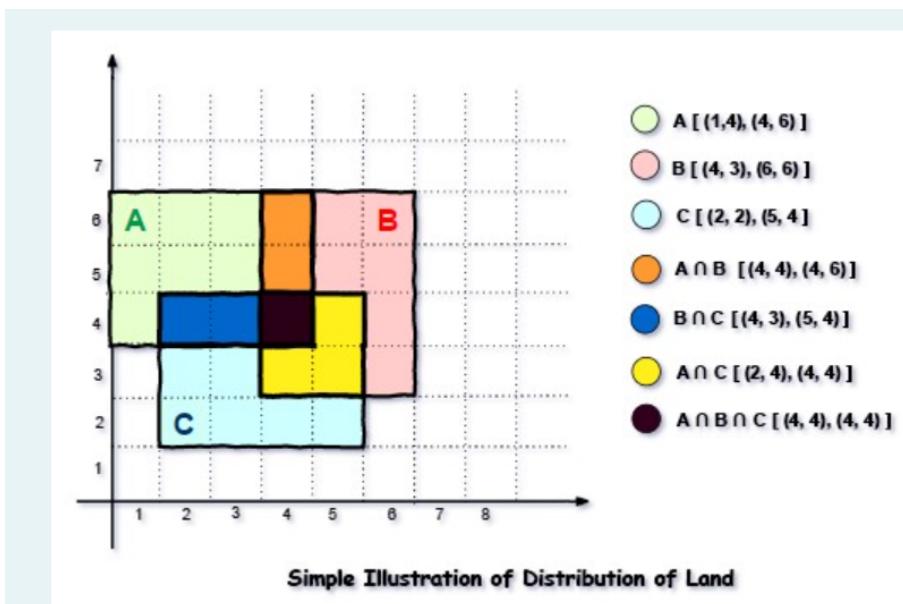
SAMPLE INPUT

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

SAMPLE OUTPUT

```
35
```

Explanation



For given sample input (see given graph for reference), compensation money for different farmers is as follows:

$$\text{Farmer with land area A: } C_1 = 5 * 1 = 5$$

$$\text{Farmer with land area B: } C_2 = 6 * 2 = 12$$

$$\text{Farmer with land area C: } C_3 = 6 * 3 = 18$$

#### **Simple Illustration of Distribution of Land**

For given sample input (see given graph for reference), compensation money for different farmers is as follows:

Farmer with land area A:  $C_1 = 5 * 1 = 5$

Farmer with land area B:  $C_2 = 6 * 2 = 12$

Farmer with land area C:  $C_3 = 6 * 3 = 18$

Total Compensation Money =  $C_1 + C_2 + C_3 = 5 + 12 + 18 = 35$

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int i,j,n,x1,x2,y1,y2,t=0;
5     long long total=0;
6     int arr[1001][1001]={0};
7     scanf("%d",&n);
8     while(n--)
9     {
10         scanf("%d %d %d %d",&x1,&y1,&x2,&y2,&t);
11         for(i=x1;i<=x2;i++)
12         {
13             for(j=y1;j<=y2;j++)
14             {
15                 if(arr[i][j]==0){
16                     arr[i][j]+=t;
17                 }
18                 else if(arr[i][j]>0){
19                     arr[i][j]=(-1)*(arr[i][j]+t);
20                 }
21                 else if(arr[i][j]<0){
22                     arr[i][j]-=t;
23                 }
24             }
25         }
26     }
```

```
12 v
13
14 v
15 v
16
17
18 v
19
20
21 v
22
23
24
25
26
27
28 v
29
30 v
31
32
33
34
35
36
37 }
```

The code is a C program. It contains two nested loops. The outer loop iterates over rows (i) from 1 to 1001. The inner loop iterates over columns (j) from 1 to 1001. For each cell (i, j), it checks the value of arr[i][j]. If the value is zero, it adds t to arr[i][j]. If the value is greater than zero, it multiplies arr[i][j] by -1 and adds t. If the value is less than zero, it subtracts t from arr[i][j]. After processing all cells, it prints the total sum of all cells multiplied by -1. Finally, it returns 0.

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 1 4 4 6 1 4 3 6 6 2 2 2 5 4 3	35	35	✓
✓	1 48 12 49 27 8	0	0	✓
✓	3 88 34 99 76 44 82 65 94 100 81 58 16 65 66 7	10500	10500	✓

Passed all tests! ✓

## Week 10 -01

Question **1**

Correct

Marked out of  
1.00

 Flag question

Given a string, **s**, consisting of alphabets and digits, find the frequency of each digit in the given string.

### **Input Format**

The first line contains a string, **num** which is the given number.

### **Constraints**

**$1 \leq \text{len}(\text{num}) \leq 1000$**

All the elements of num are made of English alphabets and digits.

### **Output Format**

Print ten space-separated integers in a single line denoting the frequency of each digit from **0** to **9**.

Print ten space-separated integers in a single line denoting the frequency of each digit from **0** to **9**.

#### **Sample Input 0**

```
a11472o5t6
```

#### **Sample Output 0**

```
0 2 1 0 1 1 1 1 0 0
```

#### **Explanation 0**

In the given string:

- **1** occurs two times.
- **2, 4, 5, 6** and **7** occur one time each.

The remaining digits **0, 3, 8** and **9** don't occur at all.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     char str[1000];
5     scanf("%s",str);
6     int hash[10]={0,0,0,0,0,0,0,0,0,0};
7     int temp;
8     for(int i=0;str[i]!='\0';i++)
9     {
10         temp=str[i]-'0';
11         if((temp<=9 && temp>=0))
12         {
13             hash[temp]++;
14         }
15     }
16     for(int i=0;i<=9;i++)
17     {
18         printf("%d ",hash[i]);
19     }
20     return 0;
21 }
22 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	a11472o5t6	0 2 1 0 1 1 1 1 0 0	0 2 1 0 1 1 1 1 0 0	✓
✓	1w4n88j12n1	0 2 1 0 1 0 0 0 2 0	0 2 1 0 1 0 0 0 2 0	✓
✓	1v88886l256338ar0ekk	1 1 1 2 0 1 2 0 5 0	1 1 1 2 0 1 2 0 5 0	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
1.00[Flag question](#)

Today, Monk went for a walk in a garden. There are many trees in the garden and each tree has an English alphabet on it. While Monk was walking, he noticed that all trees with vowels on it are not in good state. He decided to take care of them. So, he asked you to tell him the count of such trees in the garden.

**Note:** The following letters are vowels: 'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o' and 'u'.

**Input:**

The first line consists of an integer  $T$  denoting the number of test cases.

Each test case consists of only one string, each character of string denoting the alphabet (may be lowercase or uppercase) on a tree in the garden.

**Output:**

For each test case, print the count in a new line.

**Constraints:**

**$1 \leq T \leq 10$**

**$1 \leq \text{length of string} \leq 10^5$**

**SAMPLE INPUT**

2

nBBZLaosnm

JHklsnZtTL

**SAMPLE OUTPUT**

2

1

**Explanation**

In test case 1, a and o are the only vowels. So, count=2

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 #include<string.h>
3 #include<ctype.h>
4 int main()
5 {
6     int t;
7     scanf("%d",&t);
8     while(t--)
9     {
10         char str[100000];
11         int count=0;
12         scanf("%s",str);
13         //tolower(char str);
14         for(int i=0;str[i]!='\0';i++)
15         {
16             char c=str[i];
17             if((c=='a') || (c=='e') || (c=='i') || (c=='o') || (c=='u') || (c=='A') ||
18                 count++;
19         }
20         printf("%d\n",count);
21     }
22 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	2 nBBZLaosnm JHkIsnZtTL	2 1	2 1	✓
✓	2 nBBZLaosnm JHkIsnZtTL	2 1	2 1	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of  
1.00[Flag question](#)

Given a sentence,  $s$ , print each word of the sentence in a new line.

**Input Format**

The first and only line contains a sentence,  $s$ .

**Constraints**

$$1 \leq \text{len}(s) \leq 1000$$

**Output Format**

Print each word of the sentence in a new line.

**Sample Input 0**

**Sample Input 0**

This is C

**Sample Output 0**

This  
is  
C

**Explanation 0**

In the given string, there are three words ["This", "is", "C"]. We have to print each of these words in a new line.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     char s[1000];
5     scanf("%[^\\n]s",s);
6     for(int i=0;s[i]!='\\0';i++)
7     {
8         if(s[i]!=' ')
9             printf("%c",s[i]);
10        else
11            printf("\\n");
12    }
13    return 0;
14 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	This is C	This is C	This is C	✓
✓	Learning C is fun	Learning C is fun	Learning C is fun	✓

Passed all tests! ✓

**Question 4**

Correct

Marked out of  
1.00[Flag question](#)**Input Format**

You are given two strings, **a** and **b**, separated by a new line. Each string will consist of lower case Latin characters ('a'-'z').

**Output Format**

In the first line print two space-separated integers, representing the length of **a** and **b** respectively.

In the second line print the string produced by concatenating **a** and **b** (**a + b**).

In the third line print two strings separated by a space, **a'** and **b'**. **a'** and **b'** are the same as **a** and **b**, respectively, except that their first characters are swapped.

**Sample Input**

abcd

ef

**Sample Output**

### **Sample Output**

```
4 2
abcdef
ebcd af
```

### **Explanation**

```
a = "abcd"
b = "ef"
|a| = 4
|b| = 2
a + b = "abcdef"
a' = "ebcd"
b' = "af"
```

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     char str1[10],str2[10],t;
5     int i=0,j=0;
6     int count1=0,count2=0;
7     scanf("%s",str1);
8     scanf("%s",str2);
9     while(str1[i]!='\0')
10    {
11        count1++;
12        i++;
13    }
14    while(str2[j]!='\0')
15    {
16        count2++;
17        j++;
18    }
19    printf("%d %d\n",count1,count2);
20    printf("%s%s\n",str1,str2);
21    t=str1[0];
22    str1[0]=str2[0];
23    str2[0]=t;
24    printf("%s %s",str1,str2);
25    return 0;
26 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	abcd ef	4 2 abcdef ebcd af	4 2 abcdef ebcd af	✓

Passed all tests! ✓

## Week 11 - 01

Question 1

Correct

Marked out of  
1.00

 Flag question

Two strings **A** and **B** comprising of lower case English letters are compatible if they are equal or can be made equal by following this step any number of times:

- Select a prefix from the string **A** (possibly empty), and increase the alphabetical value of all the characters in the prefix by the same valid amount. For example, if the string is **xyz** and we select the prefix **xy** then we can convert it to **yx** by increasing the alphabetical value by 1. But if we select the prefix **xyz** then we cannot increase the alphabetical value.

Your task is to determine if given strings **A** and **B** are compatible.

### **Input format**

First line: String **A**

Next line: String **B**

### **Output format**

### **Output format**

For each test case, print **YES** if string **A** can be converted to string **B**, otherwise print **NO**.

Constraints

$1 \leq \text{len}(A) \leq 1000000$

$1 \leq \text{len}(B) \leq 1000000$

### **SAMPLE INPUT**

abaca

cdbda

### **SAMPLE OUTPUT**

YES

abaca

cdbda

#### SAMPLE OUTPUT

YES

#### Explanation

The string **abaca** can be converted to **bcbda** in one move and to **cdbda** in the next move.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 #include<string.h>
3 int main()
4 {
5     char str1[1000000],str2[1000000];
6     int flag=1;
7     scanf("%s",str1);
8     scanf("%s",str2);
9     int a=strlen(str1);
10    int b=strlen(str2);
11    if(a==b)
12    {
13        for(int i=a-1;i>0;i--)
14        {
15            while(str1[i]!=str2[i])
16            {
17                for(int j=0;j<=i;j++)
18                {
19                    if(str1[j]<'z')
20                        str1[j]++;
21                    else
22                    {
23                        flag=0;
24                        break;
25                    }
26                    if(flag==0)
27                        break;
```

```
20         str1[j]++;
21     else
22     {
23         flag=0;
24         break;
25     }
26     if(flag==0)
27         break;
28 }
29 }
30 }
31 }
32 else
33 flag=0;
34 if(flag==0)
35 printf("NO");
36 //printf("NO");
37 else
38 printf("YES");
39 return 0;
40 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	abaca cdbda	YES	YES	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of  
1.00

 [Flag question](#)

Danny has a possible list of passwords of Manny's facebook account. All passwords length is odd. But Danny knows that Manny is a big fan of palindromes. So, his password and reverse of his password both should be in the list.

You have to print the length of Manny's password and it's middle character.

**Note: The solution will be unique.**

### INPUT

The first line of input contains the integer N, the number of possible passwords.

Each of the following N lines contains a single word, its length being an odd number greater than 2 and lesser than **14**. All characters are lowercase letters of the English alphabet.

### OUTPUT

The first and only line of output must contain the length of the correct password and its central letter.

## **OUTPUT**

The first and only line of output must contain the length of the correct password and its central letter.

## **CONSTRAINTS**

**$1 \leq N \leq 100$**

## **SAMPLE INPUT**

```
4
abc
def
feg
cba
```

## **SAMPLE OUTPUT**

feg

cba

**SAMPLE OUTPUT**

3 b

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 #include<string.h>
3 int main()
4 {
5     int n,flag=0;
6     char temp;
7     scanf("%d",&n);
8     char words[n][14];
9     for(int i=0;i<n;i++)
10    scanf("%s",words[i]);
11    char reverse[14];
12    for(int i=0;i<n-1;i++)
13    {
14        strcpy(reverse,words[i]);
15        int size=strlen(reverse);
16
17        for(int k=0;k<size/2;k++)
18        {
19            temp=reverse[k];
20            reverse[k]=reverse[size-k-1];
21            reverse[size-k-1]=temp;
22        }
23        for(int j=i+1;j<n;j++)
24        {
25            if(strcmp(reverse,words[j])==0)
26            {
```

```
15     int size=strlen(reverse);
16
17     for(int k=0;k<size/2;k++)
18     {
19         temp=reverse[k];
20         reverse[k]=reverse[size-k-1];
21         reverse[size-k-1]=temp;
22     }
23     for(int j=i+1;j<n;j++)
24     {
25         if(strcmp(reverse,words[j])==0)
26         {
27             flag=1;
28             break;
29         }
30     }
31     if(flag==1)
32     break;
33 }
34 int len=strlen(reverse);
35 printf("%d %c ",len,reverse[len/2]);
36 return 0;
37
38 }
39 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	4 abc def feg cba	3 b	3 b	✓

Passed all tests! ✓

Question 3

Correct

Marked out of  
1.00

 Flag question

Joey loves to eat Pizza. But he is worried as the quality of pizza made by most of the restaurants is deteriorating. The last few pizzas ordered by him did not taste good :( . Joey is feeling extremely hungry and wants to eat pizza. But he is confused about the restaurant from where he should order. As always he asks Chandler for help.

Chandler suggests that Joey should give each restaurant some points, and then choose the restaurant having **maximum points**. If more than one restaurant has same points, Joey can choose the one with **lexicographically smallest** name.

Joey has assigned points to all the restaurants, but can't figure out which restaurant satisfies Chandler's criteria. Can you help him out?

**Input:**

First line has N, the total number of restaurants.

Next N lines contain Name of Restaurant and Points awarded by Joey, separated by a space. Restaurant name has **no spaces**, all lowercase letters and will not be more than 20 characters.

**Output:**

Print the name of the restaurant that Joey should choose.

**Constraints:**

$1 \leq N \leq 10^5$

$1 \leq \text{Points} \leq 10^6$

**SAMPLE INPUT**

3

Pizzeria 108

Dominos 145

Pizzapizza 49

**SAMPLE OUTPUT**

Pizzeria 108

Dominos 145

Pizzapizza 49

#### **SAMPLE OUTPUT**

Dominos

#### **Explanation**

**Dominos** has maximum points.

**Answer:** (penalty regime: 0 %)

```
1 //  
2 #include<stdio.h>  
3 #include<string.h>  
4 int main()  
5 {  
6     int n;  
7     scanf("%d",&n);  
8     char res[n][21];  
9     int rate[n];  
10    for(int i=0;i<n;i++)  
11    {  
12        scanf("%s",res[i]);  
13        scanf("%d",&rate[i]);  
14    }  
15    int max=rate[0];  
16    char ans[20];  
17    strcpy(ans,res[0]);  
18    for(int i=1;i<n;i++)  
19    {  
20        if(rate[i]>max)  
21        {  
22            max=rate[i];  
23            strcpy(ans,res[i]);  
24        }  
25        else if(rate[i]==max)  
26        {  
27            if(strcmp(res[i],ans)<0)  
.....
```

```
12     scanf("%s",res[i]),
13     scanf("%d",&rate[i]);
14 }
15 int max=rate[0];
16 char ans[20];
17 strcpy(ans,res[0]);
18 for(int i=1;i<n;i++)
19 {
20     if(rate[i]>max)
21     {
22         max=rate[i];
23         strcpy(ans,res[i]);
24     }
25     else if(rate[i]==max)
26     {
27         if(strcmp(res[i],ans)<0)
28             strcpy(ans,res[i]);
29     }
30 }
31 printf("%s",ans);
32 return 0;
33 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 Pizzeria 108 Dominos 145 Pizzapizza 49	Dominos	Dominos	✓

Passed all tests! ✓

**Question 4**

Correct

Marked out of  
1.00[Flag question](#)

These days Bechan Chacha is depressed because his crush gave him list of mobile number some of them are valid and some of them are invalid. Bechan Chacha has special power that he can pick his crush number only if he has valid set of mobile numbers. Help him to determine the valid numbers.

You are given a string "S" and you have to determine whether it is Valid mobile number or not. Mobile number is valid only if it is of length 10 , consists of numeric values and it shouldn't have prefix zeroes.

**Input:**

First line of input is T representing total number of test cases.

Next T line each representing "S" as described in in problem statement.

**Output:**

Print "YES" if it is valid mobile number else print "NO".

Note: Quotes are for clarity.

**Constraints:**

$1 \leq T \leq 10^3$

sum of string length  $\leq 10^5$

**SAMPLE INPUT**

3

1234567890

0123456789

0123456.87

**SAMPLE OUTPUT**

YES

NO

NO

```
1 #include<stdio.h>
2 #include<string.h>
3 int main()
4 {
5     int t;
6     scanf("%d",&t);
7     while(t--)
8     {
9         int flag=1;
10        char s[100000];
11        scanf("%s",s);
12        int k=strlen(s);
13        if(k==10)
14        {
15            for(int i=0;i<10;i++)
16            {
17                if(s[0]=='0')
18                {
19                    flag=0;
20                    break;
21                }
22                if(s[i]<'0' || s[i]>'9')
23                {
24                    flag=0;
25                    break;
26                }
27            }
28        }
29    }
30 }
```

```
16 v
17
18 v
19
20
21
22
23
24 v
25
26
27
28
29
30
31
32
33
34
35
36
37
38 }
```

```
    if(s[0]=='0')
    {
        flag=0;
        break;
    }
    if(s[i]<'0' || s[i]>'9')
    {
        flag=0;
        break;
    }
}
else
flag=0;
if(flag==1)
printf("YES\n");
else
printf("NO\n");
}
return 0;
}
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3	YES	YES	✓
	1234567890	NO	NO	
	0123456789	NO	NO	
	0123456.87			

Passed all tests! ✓

## Week 12 - 01

Question 1

Correct

Marked out of  
1.00

 Flag question

A binary number is a combination of 1s and 0s. Its  $n^{\text{th}}$  least significant digit is the  $n^{\text{th}}$  digit starting from the right starting with 1. Given a decimal number, convert it to binary and determine the value of the the  $4^{\text{th}}$  least significant digit.

### Example

number = 23

- Convert the decimal number 23 to binary number:  $23^{10} = 2^4 + 2^2 + 2^1 + 2^0 = (10111)_2$ .
- The value of the  $4^{\text{th}}$  index from the right in the binary representation is 0.

### Function Description

Complete the function fourthBit in the editor below.

fourthBit has the following parameter(s):

int number: a decimal integer

int number: a decimal integer

Returns:

int: an integer 0 or 1 matching the 4th least significant digit in the binary representation of number.

### Constraints

$0 \leq \text{number} < 2^{31}$

### Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The only line contains an integer, number.

### Sample Case 0

### Sample Input 0

STDIN Function

-----  
32 → number = 32

#### **Sample Output 0**

0

#### **Explanation 0**

- Convert the decimal number 32 to binary number:  $32_{10} = (100000)_2$ .
- The value of the 4th index from the right in the binary representation is 0.

#### **Sample Case 1**

#### **Sample Input 1**

**Sample Input 1**

STDIN Function

-----

77 → number = 77

**Sample Output 1**

1

**Explanation 1**

- Convert the decimal number 77 to binary number:  $77_{10} = (1001101)_2$ .
- The value of the 4th index from the right in the binary representation is 1.

```
1 /*  
2  * Complete the 'fourthBit' function below.  
3  *  
4  * The function is expected to return an INTEGER.  
5  * The function accepts INTEGER number as parameter.  
6 */  
7  
8 int fourthBit(int number)  
9 {  
10    int binary[32];  
11    int i=0;  
12    while(number>0)  
13    {  
14        binary[i]=number%2;  
15        number/=2;  
16        i++;  
17    }  
18    if(i>=4)  
19    {  
20        return binary[3];  
21    }  
22    else  
23    return 0;  
24 }
```

	Test	Expected	Got	
✓	printf("%d", fourthBit(32))	0	0	✓
✓	printf("%d", fourthBit(77))	1	1	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
1.00[Flag question](#)

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the  $p^{\text{th}}$  element of the list, sorted ascending. If there is no  $p^{\text{th}}$  element, return 0.

**Example**

$n = 20$

$p = 3$

The factors of 20 in ascending order are {1, 2, 4, 5, 10, 20}. Using 1-based indexing, if  $p = 3$ , then 4 is returned. If  $p > 6$ , 0 would be returned.

**Function Description**

Complete the function `pthFactor` in the editor below.

`pthFactor` has the following parameter(s):

int  $n$ : the integer whose factors are to be found

int  $p$ : the index of the factor to be returned

`ptnFactor` has the following parameter(s):

`int n`: the integer whose factors are to be found

`int p`: the index of the factor to be returned

Returns:

`int`: the long integer value of the  $p^{\text{th}}$  integer factor of  $n$  or, if there is no factor at that index, then 0 is returned

### Constraints

$1 \leq n \leq 10^{15}$

$1 \leq p \leq 10^9$

Input Format for Custom Testing

Input from `stdin` will be processed as follows and passed to the function.

The first line contains an integer  $n$ , the number to factor.

The second line contains an integer  $p$ , the 1-based index of the factor to return.

The first line contains an integer  $n$ , the number to factor.

The second line contains an integer  $p$ , the 1-based index of the factor to return.

**Sample Case 0**

**Sample Input 0**

STDIN    Function

-----

10    →     $n = 10$

3    →     $p = 3$

**Sample Output 0**

5

**Explanation 0**

Factoring n = 10 results in {1, 2, 5, 10}. Return the p = 3<sup>rd</sup> factor, 5, as the answer.

**Sample Case 1**

**Sample Input 1**

STDIN      Function

----- -----

10      →    n = 10

5      →    p = 5

**Sample Output 1**

0

**Explanation 1**

Factoring n = 10 results in {1, 2, 5, 10}. There are only 4 factors and p = 5, therefore 0 is returned as the answer.

**Sample Case 2****Sample Input 2**

STDIN    Function

-----

1    →   n = 1

1    →   p = 1

**Sample Output 2**

1

**Explanation 2**

Factoring n = 1 results in {1}. The p = 1st factor of 1 is returned as the answer.

**Answer:** (penalty regime: 0 %)

Reset answer

```
1  /*
2  * Complete the 'pthFactor' function below.
3  *
4  * The function is expected to return a LONG_INTEGER.
5  * The function accepts following parameters:
6  * 1. LONG_INTEGER n
7  * 2. LONG_INTEGER p
8  */
9
10 long pthFactor(long n, long p)
11 {
12     int count=0;
13     for(long i=1;i<=n;++i)
14     {
15         if(n%i==0)
16         {
17             count++;
18             if(count==p)
19             {
20                 return i;
21             }
22         }
23     }
24     return 0;
25 }
```

	Test	Expected	Got	
✓	printf("%ld", pthFactor(10, 3))	5	5	✓
✓	printf("%ld", pthFactor(10, 5))	0	0	✓
✓	printf("%ld", pthFactor(1, 1))	1	1	✓

Passed all tests! ✓

## Week 12 - 02

Question **1**

Correct

Marked out of  
1.00

 [Flag question](#)

You are a bank account hacker. Initially you have 1 rupee in your account, and you want exactly **N** rupees in your account. You wrote two hacks, first hack can multiply the amount of money you own by 10, while the second can multiply it by 20. These hacks can be used any number of time. Can you achieve the desired amount **N** using these hacks.

### Constraints:

$1 \leq T \leq 100$

$1 \leq N \leq 10^{12}$

### Input

- The test case contains a single integer N.

### Output

## **Output**

For each test case, print a single line containing the string "1" if you can make exactly N rupees or "0" otherwise.

SAMPLE INPUT

1

SAMPLE OUTPUT

1

SAMPLE INPUT

2

SAMPLE OUTPUT

```
2 // Complete the myFunc function below.
3 *
4 * The function is expected to return an INTEGER.
5 * The function accepts INTEGER n as parameter.
6 */
7
8 int myFunc(int n)
9 {
10     int sum=1,flag=0;
11     if(n==1)
12     {
13         flag=1;
14     }
15     else
16     {
17         for(int i=0;i<n;i++)
18         {
19             sum=sum*10;
20             if(sum==n)
21             {
22                 flag=1;
23                 break;
24             }
25         }
26         sum=1;
27         for(int i=0;i<n;i++)
28         {
29             sum=sum*20;
30             if(sum==n)
31             {
32                 flag=1;
```

```
25     }
26     sum=1;
27     for(int i=0;i<n;i++)
28     {
29         sum=sum*20;
30         if(sum==n)
31         {
32             flag=1;
33             break;
34         }
35     }
36     sum=1;
37     for(int i=0;i<n;i++)
38     {
39         sum=sum*10;
40         sum=sum*20;
41         if(sum==n)
42         {
43             flag=1;
44             break;
45         }
46     }
47 }
48 }
49 return(flag);
50 }
```

	Test	Expected	Got	
✓	printf("%d", myFunc(1))	1	1	✓
✓	printf("%d", myFunc(2))	0	0	✓
✓	printf("%d", myFunc(10))	1	1	✓
✓	printf("%d", myFunc(25))	0	0	✓
✓	printf("%d", myFunc(200))	1	1	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of  
1.00

 [Flag question](#)

Find the number of ways that a given integer,  $X$ , can be expressed as the sum of the  $N^{th}$  powers of unique, natural numbers.

For example, if  $X = 13$  and  $N = 2$ , we have to find all combinations of unique squares adding up to  $13$ . The only solution is  $2^2 + 3^2$ .

### Function Description

Complete the powerSum function in the editor below. It should return an integer that represents the number of possible combinations.

powerSum has the following parameter(s):

$X$ : the integer to sum to

$N$ : the integer power to raise numbers to

Input Format

N: the integer power to raise numbers to

#### Input Format

The first line contains an integer **X**.

The second line contains an integer **N**.

#### Constraints

**1 ≤ X ≤ 1000**

**2 ≤ N ≤ 10**

#### Output Format

Output a single integer, the number of possible combinations calculated.

#### Sample Input 0

10

2

2

#### **Sample Output 0**

1

#### **Explanation 0**

If  $X = 10$  and  $N = 2$ , we need to find the number of ways that  $10$  can be represented as the sum of squares of unique numbers.

$$10 = 1^2 + 3^2$$

This is the only way in which  $10$  can be expressed as the sum of unique squares.

#### **Sample Input 1**

**Sample Input 1**

100

2

**Sample Output 1**

3

**Explanation 1**

$$100 = (10^2) = (6^2 + 8^2) = (1^2 + 3^2 + 4^2 + 5^2 + 7^2)$$

**Sample Input 2**

100

3

Sample Output 2

1

### **Explanation 2**

**100** can be expressed as the sum of the cubes of **1, 2, 3, 4**.

**(1 + 8 + 27 + 64 = 100)**. There is no other way to express **100** as the sum of cubes.

**Answer:** (penalty regime: 0 %)

Reset answer

```
1  /*
2  * Complete the 'powerSum' function below.
3  *
4  * The function is expected to return an INTEGER.
5  * The function accepts following parameters:
6  * 1. INTEGER x
7  * 2. INTEGER n
8  */
9 #include<math.h>
10
11 int powerSum(int x, int m, int n)
12 {
13     int p=pow(m,n);
14     if(p==x)
15     {
16         return 1;
17     }
18     if(p>x)
19     {
20         return 0;
21     }
22     return powerSum(x-p,m+1,n)+powerSum(x,m+1,n);
23 }
```

	Test	Expected	Got	
✓	printf("%d", powerSum(10, 1, 2))	1	1	✓

Passed all tests! ✓

## Week 14 - 01

Question **1**

Correct

 [Flag question](#)

Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.

Example

arr=[1,2,3,4,6]

- the sum of the first three elements,  $1+2+3=6$ . The value of the last element is 6.
- Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
- The index of the pivot is 3.

Function Description

Complete the function balancedSum in the editor below.

balancedSum has the following parameter(s):

int arr[n]: an array of integers

`int arr[n]:` an array of integers

Returns:

`int:` an integer representing the index of the pivot

Constraints

- $3 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 2 \times 10^4$ , where  $0 \leq i < n$
- It is guaranteed that a solution always exists.

Input Format for Custom Testing

Input from `stdin` will be processed as follows and passed to the function.

The first line contains an integer  $n$ , the size of the array  $arr$ .

Each of the next  $n$  lines contains an integer,  $arr[i]$ , where  $0 \leq i < n$ .

Sample Case 0

Sample Input 0

STDIN Function Parameters

-----  
4 → arr[] size n = 4  
1 → arr = [1, 2, 3, 3]  
2  
3  
3

Sample Output 0

2

Explanation 0

- The sum of the first two elements,  $1+2=3$ . The value of the last element is 3.

- The sum of the first two elements,  $1+2=3$ . The value of the last element is 5.
- Using zero based indexing,  $\text{arr}[2]=3$  is the pivot between the two subarrays.
- The index of the pivot is 2.

Sample Case 1

Sample Input 1

STDIN    Function Parameters

-----

```
3 → arr[] size n = 3
1 → arr = [1, 2, 1]
2
1
```

Sample Output 1

1

Explanation 1

### Explanation 1

- The first and last elements are equal to 1.
- Using zero based indexing, arr[1]=2 is the pivot between the two subarrays.
- The index of the pivot is 1.

```
1 /*  
2  * Complete the 'balancedSum' function below.  
3  *  
4  * The function is expected to return an INTEGER.  
5  * The function accepts INTEGER_ARRAY arr as parameter.  
6 */  
7  
8 int balancedSum(int arr_count, int* arr)  
9 {  
10    int sum=0,lsum=0,no;  
11    for(int i=0;i<arr_count;i++)  
12    {  
13        for(int j=arr_count-1;j>=0;j--)  
14        {  
15            sum=sum+arr[i];  
16            lsum=lsum+arr[j];  
17            if(sum==lsum)  
18            {  
19                //ne=sum;  
20                no=i-1;  
21                break;  
22            }  
23        }  
24    }  
25    return no;  
26}  
27}  
28}
```

	<b>Test</b>	<b>Expected</b>	<b>Got</b>	
✓	int arr[] = {1,2,3,3}; printf("%d", balancedSum(4, arr))	2	2	✓

Passed all tests! ✓

Question **2**

Correct

 [Flag question](#)

Calculate the sum of an array of integers.

Example

`numbers = [3, 13, 4, 11, 9]`

The sum is  $3 + 13 + 4 + 11 + 9 = 40$ .

Function Description

Complete the function `arraySum` in the editor below.

`arraySum` has the following parameter(s):

`int numbers[n]: an array of integers`

Returns

`int: integer sum of the numbers array`

Returns

int: integer sum of the numbers array

Constraints

$$1 \leq n \leq 10^4$$

$$1 \leq \text{numbers}[i] \leq 10^4$$

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer  $n$ , the size of the array  $\text{numbers}$ .

Each of the next  $n$  lines contains an integer  $\text{numbers}[i]$  where  $0 \leq i < n$ .

Sample Case 0

Sample Input 0

The first line contains an integer n, the size of the array numbers.

Each of the next n lines contains an integer numbers[i] where  $0 \leq i < n$ .

Sample Case 0

Sample Input 0

STDIN      Function

----- -----

5      →    numbers[] size n = 5

1      →    numbers = [1, 2, 3, 4, 5]

2

3

4

5

Sample Output 0

15

### Explanation 0

$1 + 2 + 3 + 4 + 5 = 15.$

Sample Case 1

Sample Input 1

STDIN      Function

----- -----

2      →    numbers[] size n = 2

12     →    numbers = [12, 12]

12

Sample Output 1

24

Explanation 1

44

### Explanation 1

$$12 + 12 = 24.$$

```
1 /*  
2  * Complete the 'arraySum' function below.  
3  *  
4  * The function is expected to return an INTEGER.  
5  * The function accepts INTEGER_ARRAY numbers as parameter.  
6 */  
7  
8 int arraySum(int numbers_count, int *numbers)  
9 {  
10    int sum=0;  
11    for(int i=0;i<numbers_count;i++)  
12    {  
13        sum=sum+numbers[i];  
14    }  
15    return sum;  
16 }  
17
```

	<b>Test</b>	<b>Expected</b>	<b>Got</b>	
✓	int arr[] = {1,2,3,4,5}; printf("%d", arraySum(5, arr))	15	15	✓

Passed all tests! ✓

**Question 3**

Correct

[Flag question](#)

Given an array of  $n$  integers, rearrange them so that the sum of the absolute differences of all adjacent elements is minimized. Then, compute the sum of those absolute differences. Example  $n = 5$  arr = [1, 3, 3, 2, 4] If the list is rearranged as arr' = [1, 2, 3, 3, 4], the absolute differences are  $|1 - 2| = 1$ ,  $|2 - 3| = 1$ ,  $|3 - 3| = 0$ ,  $|3 - 4| = 1$ . The sum of those differences is  $1 + 1 + 0 + 1 = 3$ . Function Description Complete the function minDiff in the editor below. minDiff has the following parameter: arr: an integer array Returns: int: the sum of the absolute differences of adjacent elements Constraints  $2 \leq n \leq 105$   $0 \leq \text{arr}[i] \leq 109$ , where  $0 \leq i < n$  Input Format For Custom Testing The first line of input contains an integer,  $n$ , the size of arr. Each of the following  $n$  lines contains an integer that describes  $\text{arr}[i]$  (where  $0 \leq i < n$ ) . Sample Case 0 Sample Input For Custom Testing STDIN Function ----- ----- 5 → arr[] size n = 5 5 → arr[] = [5, 1, 3, 7, 3] 1 3 7 3 Sample Output 6 Explanation n = 5 arr = [5, 1, 3, 7, 3] If arr is rearranged as arr' = [1, 3, 3, 5, 7], the differences are minimized. The final answer is  $|1 - 3| + |3 - 3| + |3 - 5| + |5 - 7| = 6$ . Sample Case 1 Sample Input For Custom Testing STDIN Function ----- ----- 2 → arr[] size n = 2 3 → arr[] = [3, 2] 2 Sample Output 1 Explanation n = 2 arr = [3, 2] There is no need to rearrange because there are only two elements. The final answer is  $|3 - 2| = 1$ .

**Answer:** (penalty regime: 0 %)

Reset answer

```
1 #include<stdlib.h>
2 /*
3 *
4 * Complete the 'minDiff' function below.
5 *
6 * The function is expected to return an INTEGER.
7 * The function accepts INTEGER_ARRAY arr as parameter.
8 */
9
10 int minDiff(int arr_count, int* arr)
11 {
12     int temp,j,sum=0;
13     for(int i=0;i<arr_count-1;i++)
14     {
15         for(j=0;j<arr_count-i-1;j++)
16         {
17             if(arr[j]>arr[j+1])
18             {
19                 temp=arr[j];
20                 arr[j]=arr[j+1];
21                 arr[j+1]=temp;
22             }
23         }
24     }
```

```
25 ▼   for(int i=0;i<arr_count-1;i++){
26     //printf("%d",arr[i]);
27     sum+=abs(arr[i]-arr[i+1]);
28   }
29   //sum=sum/2;
30   return sum;
31 }
32 }
```

	Test	Expected	Got	
✓	int arr[] = {5, 1, 3, 7, 3}; printf("%d", minDiff(5, arr))	6	6	✓

Passed all tests! ✓

## Week 14 - 01

Question 1

Correct

 [Flag question](#)

You are transporting some boxes through a tunnel, where each box is a parallelepiped, and is characterized by its length, width and height.

The height of the tunnel **41** feet and the width can be assumed to be infinite. A box can be carried through the tunnel only if its height is strictly less than the tunnel's height. Find the volume of each box that can be successfully transported to the other end of the tunnel. Note: Boxes cannot be rotated.

Input Format

The first line contains a single integer ***n***, denoting the number of boxes.

***n*** lines follow with three integers on each separated by single spaces - ***length<sub>i</sub>***, ***width<sub>i</sub>*** and ***height<sub>i</sub>*** which are length, width and height in feet of the ***i*-th** box.

Constraints

**$1 \leq n \leq 100$**

**$1 \leq \text{length}_i, \text{width}_i, \text{height}_i \leq 100$**

#### Output Format

For every box from the input which has a height lesser than **41** feet, print its volume in a separate line.

#### Sample Input 0

```
4
5 5 5
1 2 40
10 5 41
7 2 42
```

#### Sample Output 0

```
125
80
```

#### Sample Output 0

125

80

#### Explanation 0

The first box is really low, only **5** feet tall, so it can pass through the tunnel and its volume is  **$5 \times 5 \times 5 = 125$** .

The second box is sufficiently low, its volume is  **$1 \times 2 \times 4 = 80$** .

The third box is exactly **41** feet tall, so it cannot pass. The same can be said about the fourth box.

**Answer:** (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int t,i=0;
5     scanf("%d",&t);
6     while(i<t)
7     {
8         int h,l,w,vol;
9         scanf("%d %d %d",&l,&w,&h);
10        if(h<41)
11        {
12            vol=h*l*w;
13            printf("%d\n",vol);
14        }
15        i++;
16    }
17 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	4	125	125	✓
	5 5 5	80	80	
	1 2 40			
	10 5 41			
	7 2 42			

Passed all tests! ✓

Question **2**

Correct

 [Flag question](#)

You are given  $n$  triangles, specifically, their sides  $a_i$ ,  $b_i$  and  $c_i$ . Print them in the same style but sorted by their areas from the smallest one to the largest one. It is guaranteed that all the areas are different.

The best way to calculate a volume of the triangle with sides  $a$ ,  $b$  and  $c$  is Heron's formula:

$$S = \sqrt{p * (p - a) * (p - b) * (p - c)} \text{ where } p = (a + b + c) / 2.$$

Input Format

First line of each test file contains a single integer  $n$ .  $n$  lines follow with  $a_i$ ,  $b_i$  and  $c_i$  on each separated by single spaces.

Constraints

$$1 \leq n \leq 100$$

$$1 \leq a_i, b_i, c_i \leq 70$$

$$a_i + b_i > c_i, a_i + c_i > b_i \text{ and } b_i + c_i > a_i$$

$a_i + b_i > c_i$ ,  $a_i + c_i > b_i$  and  $b_i + c_i > a_i$

#### Output Format

Print exactly  $n$  lines. On each line print 3 integers separated by single spaces, which are  $a_i$ ,  $b_i$  and  $c_i$  of the corresponding triangle.

#### Sample Input 0

```
3
7 24 25
5 12 13
3 4 5
```

#### Sample Output 0

```
3 4 5
5 12 13
```

3 4 5

#### Sample Output 0

3 4 5

5 12 13

7 24 25

#### Explanation 0

The square of the first triangle is **84**. The square of the second triangle is **30**. The square of the third triangle is **6**. So the sorted order is the reverse one.

```
1 #include<stdio.h>
2 #include<math.h>
3
4 typedef struct
5 {
6     int a;
7     int b;
8     int c;
9     int area;
10 }Triangle;
11
12 int area(int a,int b,int c)
13 {
14     int p=(a+b+c)/3;
15     return sqrt(p*(p-a)*(p-b)*(p-c));
16 }
17
18 void sort(Triangle triangles[],int n)
19 {
20     for(int i=0;i<n-1;i++)
21     {
22         for(int j=0;j<n-i-1;j++)
23         {
24             if(triangles[j].area>triangles[j+1].area)
25             {
26                 Triangle temp=triangles[j];
27                 triangles[j]=triangles[j+1];
28                 triangles[j+1]=temp;
29             }
30         }
31     }
32 }
```

```
27         triangles[j]=triangles[j+1];
28         triangles[j+1]=temp;
29     }
30 }
31 }
32 }
33 int main()
34 {
35     int n;
36     scanf("%d",&n);
37     Triangle triangles[n];
38     for(int i=0;i<n;i++)
39     {
40         int a,b,c;
41         scanf("%d %d %d",&a,&b,&c);
42         triangles[i].a=a;
43         triangles[i].b=b;
44         triangles[i].c=c;
45         triangles[i].area=area(a,b,c);
46     }
47     sort(triangles,n);
48     for(int i=0;i<n;i++)
49     {
50         printf("%d %d %d\n",triangles[i].a,triangles[i].b,triangles[i].c);
51     }
52     return 0;
53 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 7 24 25 5 12 13 3 4 5	3 4 5 5 12 13 7 24 25 3 4 5	3 4 5 5 12 13 7 24 25 3 4 5	✓

Passed all tests! ✓

# Week 15 - 01

Question 1

Correct

Marked out of  
1.00

 Flag question

Given an array of integers, reverse the given array in place using an index and loop rather than a built-in function.

## Example

*arr = [1, 3, 2, 4, 5]*

Return the array *[5, 4, 2, 3, 1]* which is the reverse of the input array.

## Function Description

Complete the function *reverseArray* in the editor below.

*reverseArray* has the following parameter(s):

*int arr[n]:* an array of integers

Return

*int[n]:* the array in reverse order

## Constraints

$1 \leq n \leq 100$

$0 < arr[i] \leq 100$

## Input Format For Custom Testing

The first line contains an integer, *n*, the number of elements in *arr*.

Each line *i* of the *n* subsequent lines (where  $0 < i < n$ ) contains an integer, *arr[i]*.

Each line  $i$  of the  $n$  subsequent lines (where  $0 \leq i < n$ ) contains an integer,  $arr[i]$ .

### **Sample Case 0**

#### **Sample Input For Custom Testing**

```
5  
1  
3  
2  
4  
5
```

#### **Sample Output**

```
5  
4  
2  
3  
1
```

#### **Explanation**

The input array is [1, 3, 2, 4, 5], so the reverse of the input array is [5, 4, 2, 3, 1].

### **Sample Case 1**

**Sample Input For Custom Testing**

4

17

10

21

45

**Sample Output**

45

21

10

17

**Explanation**

The input array is [17, 10, 21, 45], so the reverse of the input array is [45, 21, 10, 17].

**Answer:** (penalty regime: 0 %)

Reset answer

```
1  /*
2  * Complete the 'reverseArray' function below.
3  *
4  * The function is expected to return an INTEGER_ARRAY.
5  * The function accepts INTEGER_ARRAY arr as parameter.
6  */
7
8  /*
9  * To return the integer array from the function, you should:
10 *     - Store the size of the array to be returned in the result_count variable
11 *     - Allocate the array statically or dynamically
12 *
13 * For example,
14 * int* return_integer_array_using_static_allocation(int* result_count) {
15 *     *result_count = 5;
16 *
17 *     static int a[5] = {1, 2, 3, 4, 5};
18 *
19 *     return a;
20 * }
21 *
22 * int* return_integer_array_using_dynamic_allocation(int* result_count) {
23 *     *result_count = 5;
24 *
25 *     int *a = malloc(5 * sizeof(int));
26 *
```

```
--  
19 *     return a;  
20 * }  
21 *  
22 * int* return_integer_array_using_dynamic_allocation(int* result_count) {  
23 *     *result_count = 5;  
24 *  
25 *     int *a = malloc(5 * sizeof(int));  
26 *  
27 *     for (int i = 0; i < 5; i++) {  
28 *         *(a + i) = i + 1;  
29 *     }  
30 *  
31 *     return a;  
32 * }  
33 *  
34 */  
35 int* reverseArray(int arr_count, int *arr, int *result_count) {  
36     *result_count = arr_count;  
37     int* reversed_arr=(int*)malloc(arr_count*sizeof(int));  
38     for(int i=0;i<arr_count;i++)  
39     {  
40         reversed_arr[i]=arr[arr_count - i-1];  
41     }  
42     return reversed_arr;  
43 }  
44 }  
45 }
```

	<b>Test</b>	<b>Expected</b>	<b>Got</b>	
✓	int arr[] = {1, 3, 2, 4, 5}; int result_count; int* result = reverseArray(5, arr, &result_count); for (int i = 0; i < result_count; i++) printf("%d\n", *(result + i));	5 4 2 3 1	5 4 2 3 1	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of  
1.00 Flag question

An automated cutting machine is used to cut rods into segments. The cutting machine can only hold a rod of *minLength* or more, and it can only make one cut at a time. Given the array *lengths[]* representing the desired lengths of each segment, determine if it is possible to make the necessary cuts using this machine. The rod is marked into lengths already, in the order given.

**Example** $n = 3$  $\text{lengths} = [4, 3, 2]$  $\text{minLength} = 7$ 

The rod is initially  $\text{sum}(\text{lengths}) = 4 + 3 + 2 = 9$  units long. First cut off the segment of length  $4 + 3 = 7$  leaving a rod  $9 - 7 = 2$ . Then check that the length 7 rod can be cut into segments of lengths 4 and 3. Since 7 is greater than or equal to  $\text{minLength} = 7$ , the final cut can be made. Return "Possible".

**Example** $n = 3$

*n* = 3

*lengths* = [4, 2, 3]

*minLength* = 7

The rod is initially  $\text{sum}(\text{lengths}) = 4 + 2 + 3 = 9$  units long. In this case, the initial cut can be of length 4 or  $4 + 2 = 6$ . Regardless of the length of the first cut, the remaining piece will be shorter than *minLength*. Because  $n - 1 = 2$  cuts cannot be made, the answer is "*Impossible*".

## Function Description

Complete the function *cutThemAll* in the editor below.

*cutThemAll* has the following parameter(s):

*int lengths[n]:* the lengths of the segments, in order

*int minLength:* the minimum length the machine can accept

Returns

## Returns

string: "*Possible*" if all  $n-1$  cuts can be made. Otherwise, return the string "*Impossible*".

## Constraints

- $2 \leq n \leq 10^5$
- $1 \leq t \leq 10^9$
- $1 \leq \text{lengths}[i] \leq 10^9$
- *The sum of the elements of lengths equals the uncut rod length.*

## **Input Format For Custom Testing**

The first line contains an integer,  $n$ , the number of elements in *lengths*.

Each line  $i$  of the  $n$  subsequent lines (where  $0 \leq i < n$ ) contains an integer, *lengths[i]*.

The next line contains an integer, *minLength*, the minimum length accepted by the machine.

The next line contains an integer, *minLength*, the minimum length accepted by the machine.

### Sample Case 0

#### Sample Input For Custom Testing

STDIN    Function

----- -----

4 → lengths[] size n = 4

3 → lengths[] = [3, 5, 4, 3]

5

4

3

9 → minLength= 9

#### Sample Output

Possible

### Explanation

The uncut rod is  $3 + 5 + 4 + 3 = 15$  units long. Cut the rod into lengths of  $3 + 5 + 4 = 12$  and 3. Then cut the 12 unit piece into lengths 3 and  $5 + 4 = 9$ . The remaining segment is  $5 + 4 = 9$  units and that is long enough to make the final cut.

### Sample Case 1

#### Sample Input For Custom Testing

STDIN    Function

----- -----

3 → lengths[] size n = 3

5 → lengths[] = [5, 6, 2]

6

2

2

12 → minLength= 12

### Sample Output

Impossible

### Explanation

The uncut rod is  $5 + 6 + 2 = 13$  units long. After making either cut, the rod will be too short to make the second cut.

**Answer:** (penalty regime: 0 %)

Reset answer

```
1  /*
2  * Complete the 'cutThemAll' function below.
3  *
4  * The function is expected to return a STRING.
5  * The function accepts following parameters:
6  * 1. LONG_INTEGER_ARRAY lengths
7  * 2. LONG_INTEGER minLength
8  */
9
10 /*
11 * To return the string from the function, you should either do static allocation or dyna
12 *
13 * For example,
14 * char* return_string_using_static_allocation() {
15 *     static char s[] = "static allocation of string";
16 *
17 *     return s;
18 * }
19 *
20 * char* return_string_using_dynamic_allocation() {
21 *     char* s = malloc(100 * sizeof(char));
22 *
23 *     s = "dynamic allocation of string";
24 *
```

```
21 *     char* s = malloc(100 * sizeof(char));
22 *
23 *     s = "dynamic allocation of string";
24 *
25 *     return s;
26 */
27 *
28 */
29 v char* cutThemAll(int lengths_count, long *lengths, long minLength) {
30     long t=0,i=1;
31 v     for(int i=0;i<=lengths_count-1;i++){
32         t+=lengths[i];
33     }
34 v     do{
35 v         if(t-lengths[lengths_count-i-1]<minLength){
36             return "Impossible";
37         }
38         i++;
39     }while(i<lengths_count-1);
40     return "Possible";
41
42 }
43
```

	<b>Test</b>	<b>Expected</b>	<b>Got</b>	
✓	long lengths[] = {3, 5, 4, 3}; printf("%s", cutThemAll(4, lengths, 9))	Possible	Possible	✓
✓	long lengths[] = {5, 6, 2}; printf("%s", cutThemAll(3, lengths, 12))	Impossible	Impossible	✓

Passed all tests! ✓