RAJALAKSHMI
ENGINEERING COLLEGE
RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23221

PYTHON PROGRAMMING LAB

Laboratory Observation Note Book

NAME: S.Kumaran

YEAR/BRANCH/SECTION: I ST YEAR/ CSE / C-SECTION

REGISTER NO:- 2116230701159

SEMESTER: 2ND SEMESTER

ACADEMIC YEAR:- 2023-2024

Week 01 - Introduction to Python-Variables-Datatypes

Input/Output-Formatting

1.1 Converting Input Strings

Write a program to convert strings to an integer and float and display its type.

Sample Output:

10,<class 'int'>

10.9,<class 'float'>

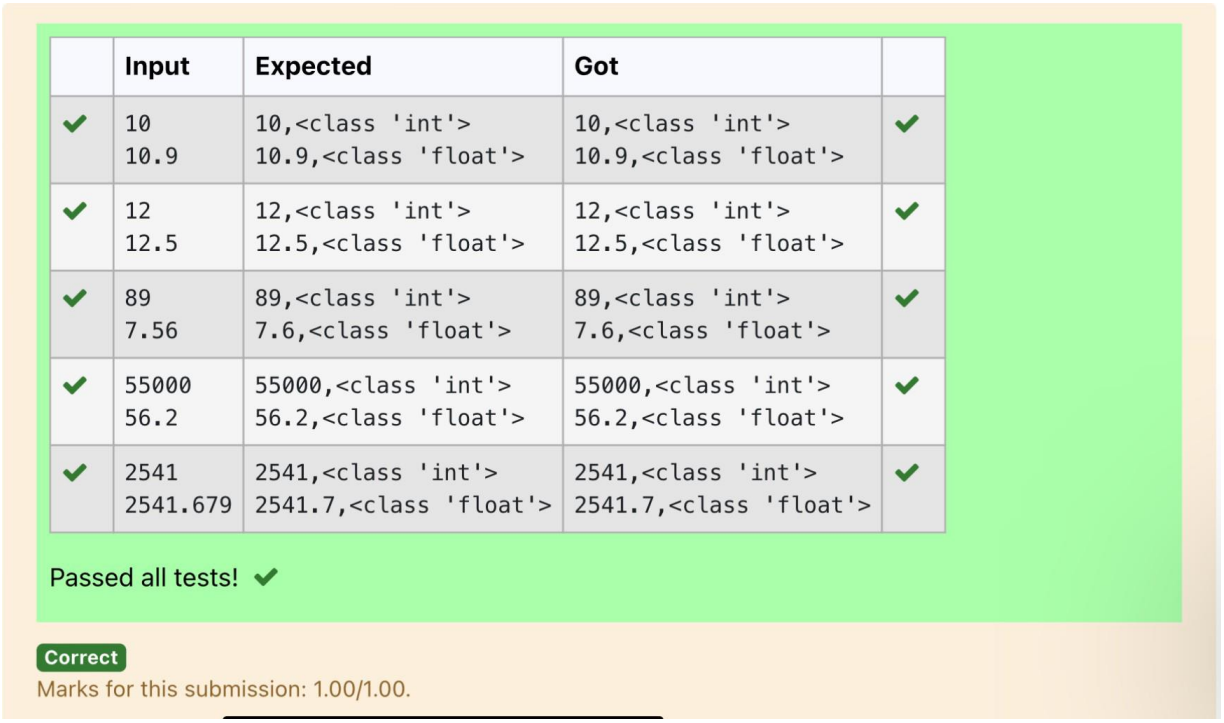
For example:

Input	**Result**
10 10.9	10,<class 'int'> 10.9,<class 'float'>

Program**:**

```
a=int(input())
b=float(input())
print(a,type(a),sep=',')
print(round(b,1),type(b),sep=',')
```

Output:



	Input	Expected	Got	
✔	10 10.9	10,<class 'int'> 10.9,<class 'float'>	10,<class 'int'> 10.9,<class 'float'>	✔
✔	12 12.5	12,<class 'int'> 12.5,<class 'float'>	12,<class 'int'> 12.5,<class 'float'>	✔
✔	89 7.56	89,<class 'int'> 7.6,<class 'float'>	89,<class 'int'> 7.6,<class 'float'>	✔
✔	55000 56.2	55000,<class 'int'> 56.2,<class 'float'>	55000,<class 'int'> 56.2,<class 'float'>	✔
✔	2541 2541.679	2541,<class 'int'> 2541.7,<class 'float'>	2541,<class 'int'> 2541.7,<class 'float'>	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

1.2 Gross Salary

Ramesh's basic salary is input through the keyboard. His dearness allowance is 40% of his basic salary, and his house rent allowance is 20% of his basic salary. Write a program to calculate his gross salary.

Sample Input:

10000

Sample Output:

16000

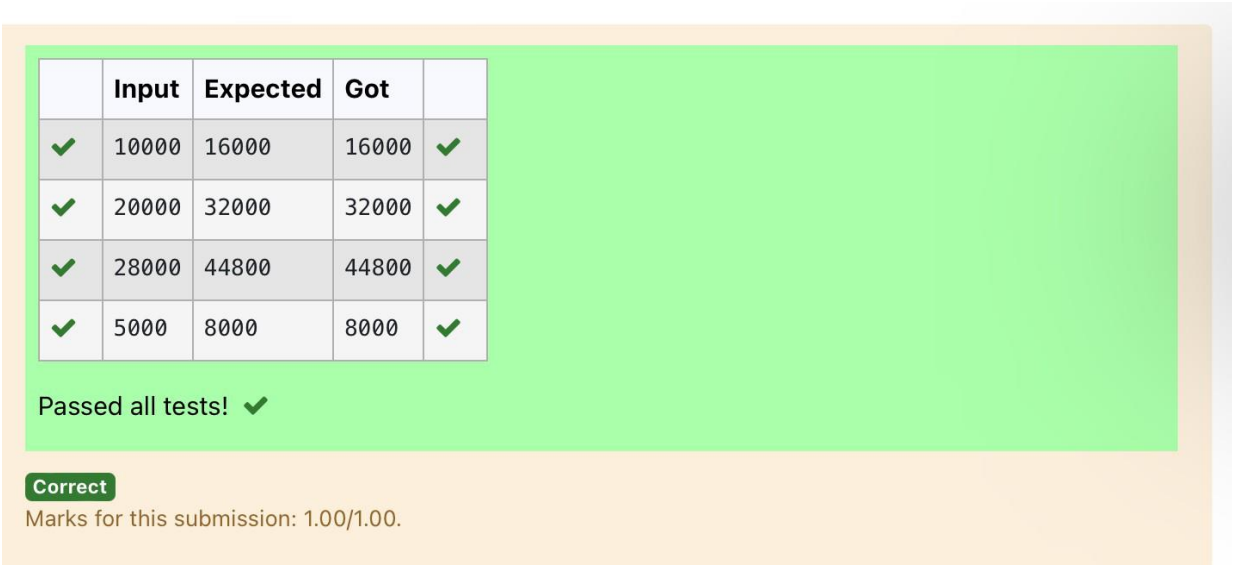
For example:

Input	Result
10000	16000

Program:

```
sal=int(input())
d_all=(40/100)*sal
houserent=(20/100)*sal
sal=sal+d_all+houserent
print(round(sal))
```

Output:



	Input	Expected	Got	
✔	10000	16000	16000	✔
✔	20000	32000	32000	✔
✔	28000	44800	44800	✔
✔	5000	8000	8000	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

1.3 Square Root

Write a simple python program to find the square root of a given floating point number. The output should be displayed with 3 decimal places.

Sample Input:

8.00

Sample Output:

2.828

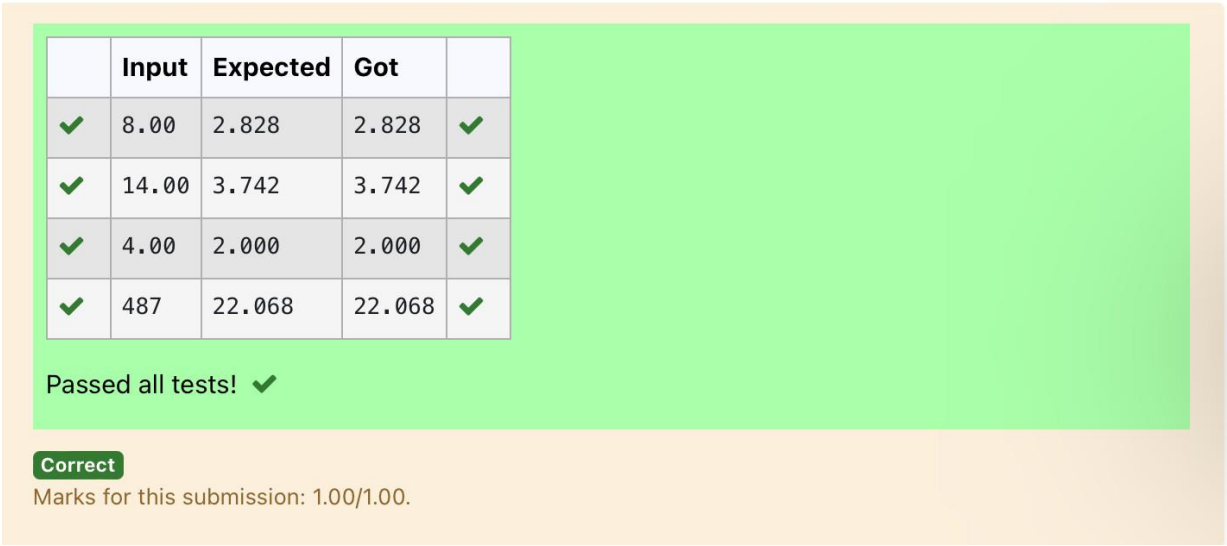
For example:

Input	Result
14.00	3.742

Program**:**

```
a=float(input())
print("%.3f"%a**0.5)
```

Output:



	Input	Expected	Got	
✔	8.00	2.828	2.828	✔
✔	14.00	3.742	3.742	✔
✔	4.00	2.000	2.000	✔
✔	487	22.068	22.068	✔

Passed all tests! ✔

1.4 Gain percent

Alfred buys an old scooter for Rs. X and spends Rs. Y on its repairs. If he sells the scooter for Rs. Z (Z>X+Y). Write a program to help Alfred to find his gain percent. Get all the above-mentioned values through the keyboard and find the gain percent.

Input Format:

The first line contains the Rs X The

second line contains Rs Y The third

line contains Rs Z Sample Input:

10000

250

15000

Sample Output:

46.34 is the gain percent.

For example:

Input	Result
45500 500 60000	30.43 is the gain percent.

Program:

```
a=int(input())
b=int(input())
c=int(input())
g=c-(a+b)
p=float((g*100)/(a+b))
print("%.2f is the gain percent."%p)
```

Output:

	Input	Expected	Got	
✔	10000 250 15000	46.34 is the gain percent.	46.34 is the gain percent.	✔
✔	45500 500 60000	30.43 is the gain percent.	30.43 is the gain percent.	✔
✔	5000 0 7000	40.00 is the gain percent.	40.00 is the gain percent.	✔
✔	12500 5000 18000	2.86 is the gain percent.	2.86 is the gain percent.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

1.5 Deposits

In many jurisdictions, a small deposit is added to drink containers to encourage people to recycle them. In one particular jurisdiction, drink containers holding one liter or less have a $0.10 deposit and drink containers holding more than one liter have a $0.25 deposit. Write a program that reads the number of containers of each size(less and more) from the user. Your program should continue by computing and displaying the refund that will be received for returning those containers. Format the output so that it includes a dollar sign and always displays exactly two decimal places.

Sample Input:

10

20

Sample Output

Your total refund will be $6.00.

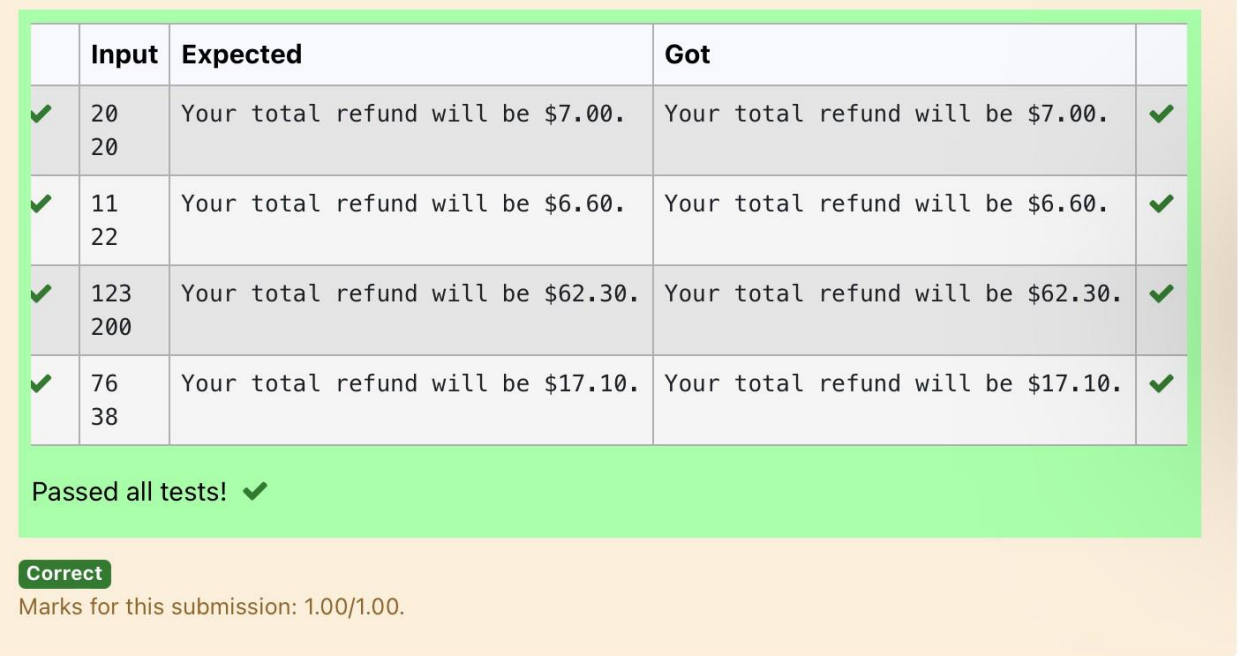
For example:

Input	Result
20 20	Your total refund will be $7.00.

Program:

```
a=float(input())
b=float(input())
h=a*0.1
j=b*0.25
k=h+j
print("Your total refund will be $%.2f."%k)
```

Output:



	Input	Expected	Got	
✔	20 20	Your total refund will be $7.00.	Your total refund will be $7.00.	✔
✔	11 22	Your total refund will be $6.60.	Your total refund will be $6.60.	✔
✔	123 200	Your total refund will be $62.30.	Your total refund will be $62.30.	✔
✔	76 38	Your total refund will be $17.10.	Your total refund will be $17.10.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

1.6 Carpenter

Justin is a carpenter who works on an hourly basis. He works in a company where he is paid Rs 50 for an hour on weekdays and Rs 80 for an hour on weekends. He works 10 hrs more on weekdays than weekends. If the salary paid for him is given, write a program to find the number of hours he has worked on weekdays and weekends.

Hint:

If the final result(hrs) are in -ve convert that to +ve using abs() function The abs()

function returns the absolute value of the given number.

```
number = -20
absolute_number = abs(number)
print(absolute_number)
# Output:20
```

Sample Input:

450

Sample Output:

weekdays 10.38

weekend 0.38

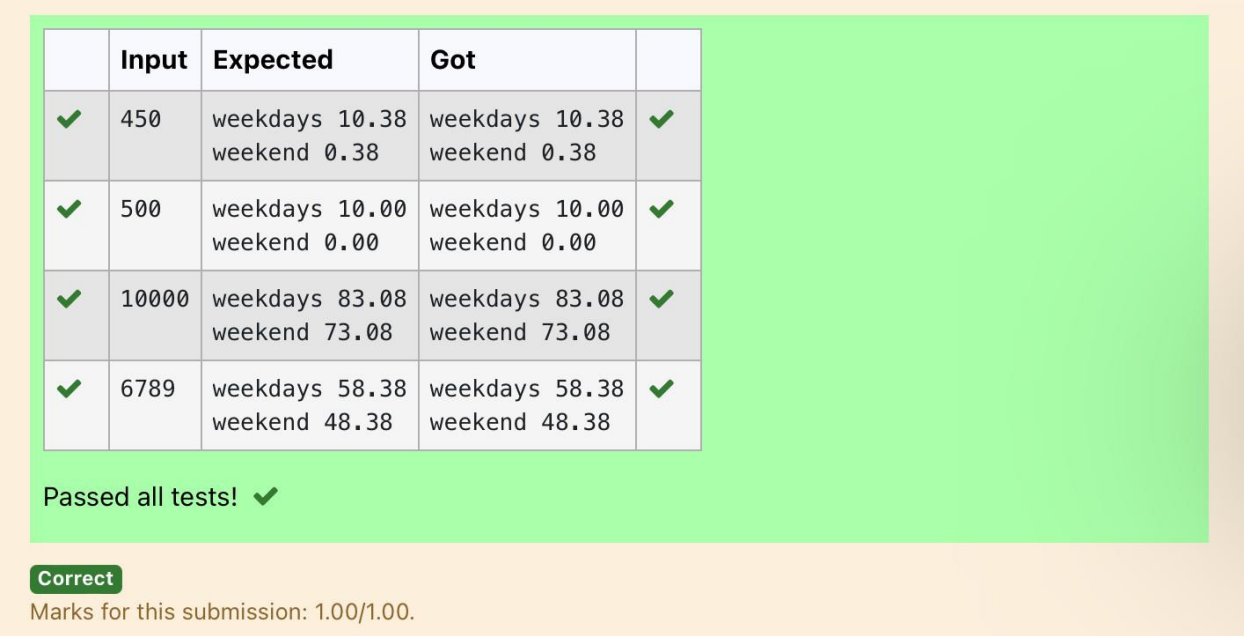
For example:

Input	Result
450	weekdays 10.38 weekend 0.38

Program:

```
num=int(input())
hours=abs((num-500)/130)
print("weekdays %.2f"%(hours+10),"\nweekend %.2f"%hours)
```

Output:



	Input	Expected	Got	
✔	450	weekdays 10.38 weekend 0.38	weekdays 10.38 weekend 0.38	✔
✔	500	weekdays 10.00 weekend 0.00	weekdays 10.00 weekend 0.00	✔
✔	10000	weekdays 83.08 weekend 73.08	weekdays 83.08 weekend 73.08	✔
✔	6789	weekdays 58.38 weekend 48.38	weekdays 58.38 weekend 48.38	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

-02- Operators in Python

2.1

Widgets and Gizmos

An online retailer sells two products: widgets and gizmos. Each widget weighs 75 grams. Each gizmo weighs 112 grams. Write a program that reads the number of widgets and the number of gizmos from the user. Then your program should compute and display the total weight of the parts.

Sample Input 10

20

Sample Output

The total weight of all these widgets and gizmos is 2990 grams.

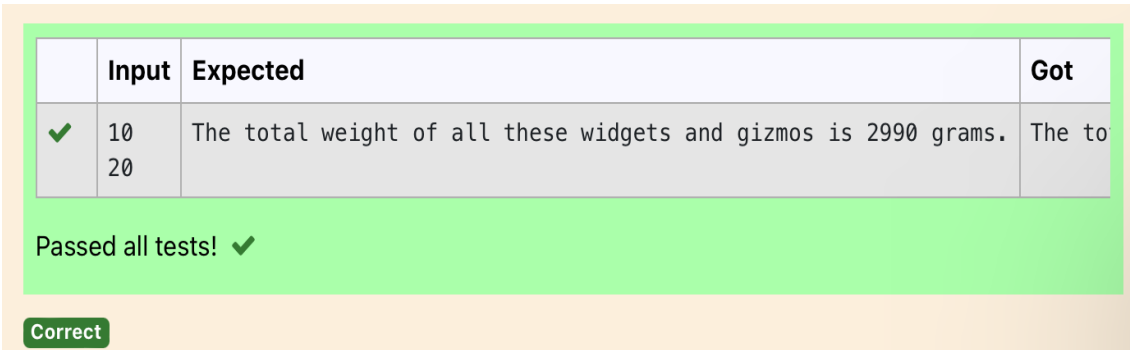
For example:

Input	Result
10 20	The total weight of all these widgets and gizmos is 2990 grams.

Program:

```
print("The total weight of all these widgets and gizmos is",(int (input())*75)+(int(input())*112),"grams.")
```

Output:



	Input	Expected	Got
✔	10 20	The total weight of all these widgets and gizmos is 2990 grams.	The to

Passed all tests! ✔

Correct

2.2

Doll Sings

In London, every year during Dasara there will be a very grand doll show. People try to invent new dolls of different varieties. The best-sold doll's creator will be awarded with a cash prize. So people broke their heads to create dolls innovatively. Knowing this competition, Mr.Lokpaul tried to create a doll that sings only when an even number is pressed and the number should not be zero and greater than 100.

IF Lokpaul wins print true, otherwise false.

Sample Input

10

Sample Output

True

Explanation:

Since 10 is an even number and a number between 0 and 100, True is printed

Program:

```
a=int(input())
if(a%2==0 and a>0 and a<100):
    print(True)
else:
    print(False)
```

Output:

	Input	Expected	Got	
✔	56	True	True	✔
✔	101	False	False	✔
✔	-1	False	False	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.3

Birthday Party

Mr. X's birthday is in next month. This time he is planning to invite N of his friends. He wants to distribute some chocolates to all of his friends after the party. He went to a shop to buy a packet of chocolates. At the chocolate shop, 4 packets are there with different numbers of chocolates. He wants to buy such a packet which contains a number of chocolates, which can be distributed equally among all of his friends. Help Mr. X to buy such a packet.

Input Given:

N-No of friends

P1,P2,P3 AND P4-No of chocolates

OUTPUT:

"True" if he can buy that packet and "False" if he can't buy that packet. SAMPLE

INPUT AND OUTPUT:

5

25

12

10

9

OUTPUT

True False True False

Program:

```
a=int(input())
b=int(input())
c=int(input())
d=int(input())
e=int(input())
print(b%a==0,c%a==0,d%a==0,e%a==0)
```

Output:

	Input	Expected	Got	
✔	5 25 23 20 10	True False True True	True False True True	✔
✔	4 23 24 21 12	False True False True	False True False True	✔
✔	8 64 8 16 32	True True True True	True True True True	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.4

Hamming Weight

Write a python program that takes a integer between 0 and 15 as input and displays the number of '1' s in its binary form.(Hint:use python bitwise operator.

Sample Input 3

Sample Output:

2

Explanation:

The binary representation of 3 is 011, hence there are 2 ones in it. so the output is 2.

Program:

```
num=int(input())

binary = bin(num)

x = [ones for ones in binary[2:] if ones=='1']

print(len(x))
```

Output:

	Input	Expected	Got	
✔	3	2	2	✔
✔	5	2	2	✔
✔	15	4	4	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.5

Compound Interest

Pretend that you have just opened a new savings account that earns 4 percent interest per year. The interest that you earn is paid at the end of the year, and is added to the balance of the savings account. Write a program that begins by reading the amount of money deposited into the account from the user. Then your program should compute and display the amount in the savings account after 1, 2, and 3 years. Display each amount so that it is rounded to 2 decimal places.

Sample Input:

10000

Sample Output:

Balance as of end of Year 1: $10400.00.

Balance as of end of Year 2: $10816.00.

Balance as of end of Year 3: $11248.64

Program:

```
a=int(input())

b=(a*0.04)+a

c=b+(b*0.04)

d=c+(c*0.04)


print("Balance as of end of Year 1: ${:.2f}.".format(b))

print("Balance as of end of Year 2: ${:.2f}.".format(c))

print("Balance as of end of Year 3: ${:.2f}.".format(d))
```

Output:

	Input	Expected	Got
✔	10000	Balance as of end of Year 1: $10400.00. Balance as of end of Year 2: $10816.00. Balance as of end of Year 3: $11248.64.	Balance as of end of Year 1: $ Balance as of end of Year 2: $ Balance as of end of Year 3: $
✔	20000	Balance as of end of Year 1: $20800.00. Balance as of end of Year 2: $21632.00. Balance as of end of Year 3: $22497.28.	Balance as of end of Year 1: $ Balance as of end of Year 2: $ Balance as of end of Year 3: $

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.6

Eligible to donate blood

A team from the Rotract club had planned to conduct a rally to create awareness among the Coimbatore people to donate blood. They conducted the rally successfully. Many of the Coimbatore people realized it and came forward to donate their blood to nearby blood banks. The eligibility criteria for donating blood are people should be above or equal to 18 and his/ her weight should be above 40. There was a huge crowd and staff in the blood bank found it difficult to manage the crowd. So they decided to keep a system and ask the people to enter their age and weight in the system. If a person is eligible he/she will be allowed inside.

Write a program and feed it to the system to find whether a person is eligible or not.

Input Format:

Input consists of two integers that correspond to the age and weight of a person respectively.

Output Format:

Display True(IF ELIGIBLE)

Display False (if not eligible)

Sample Input

19

45

Sample Output

True

Program:

```
a=int(input())

b=int(input())

if(a>=18 and b>40):

    print("True") else:

    print("False")
```

Output:

	Input	Expected	Got	
✔	19 45	True	True	✔
✔	18 40	False	False	✔
✔	18 42	True	True	✔
✔	16 45	False	False	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.7

C or D

Mr.Ram has been given a problem kindly help him to solve it. The input of the program is either 0 or 1. IF 0 is the input he should display "C" if 1 is the input it should display "D". There is a constraint that Mr. Ram should use either logical operators or arithmetic operators to solve the problem, not anything else.

Hint:

Use ASCII values of C and D.

Input Format:

An integer x, 0<=x<=1.

Output Format:

output a single character "C" or "D"depending on the value of x.

Input 1:

0

Output 1:

C

Input 2:

1

Output 1:

D

Program:

```
a=int(input()) if(a==0):
    print("C") else:
    print("D")
```

Output:

	Input	Expected	Got	
✔	0	C	C	✔
✔	1	D	D	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.8

Troy Battle

In the 1800s, the battle of Troy was led by Hercules. He was a superstitious person. He believed that his crew can win the battle only if the total count of the weapons in hand is in multiple of 3 and the soldiers are in an even number of count. Given the total number of weapons and the soldier's count, Find whether the battle can be won or not according to Hercules's belief. If the battle can be won print True otherwise print False.

Input format:

Line 1 has the total number of weapons Line

2 has the total number of Soldiers. **Output**

Format:

If the battle can be won print True otherwise print False.

Sample Input:

32

43

Sample Output:' False

Program:

```
a=int(input())

b=int(input())

if(a%3==0 and b%2==0): print("True")

else:

    print("False")
```

Output:

	Input	Expected	Got	
✔	32 43	False	False	✔
✔	273 7890	True	True	✔
✔	800 4590	False	False	✔
✔	6789 32996	True	True	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.9

Tax and Tip

The program that you create for this exercise will begin by reading the cost of a meal ordered at a restaurant from the user. Then your program will compute the tax and tip for the meal. Use your local tax rate (5 percent) when computing the amount of tax owing. Compute the tip as 18 percent of the meal amount (without the tax). The output from your program should include the tax amount, the tip amount, and the grand total for the meal including both the tax and the tip. Format the output so that all of the values are displayed using two decimal places.

Sample Input

100

Sample Output

The tax is 5.00 and the tip is 18.00, making the total 123.00

Program:

```
a=int(input())
print("The tax is {:.2f} and the tip is {:.2f}, making the total
{:.2f}".format((a*0.05),(a*0.18),(a+((a*0.05)+(a*0.18)))))
```

Output:

	Input	Expected	Got
✔	100	The tax is 5.00 and the tip is 18.00, making the total 123.00	The tax
✔	250	The tax is 12.50 and the tip is 45.00, making the total 307.50	The tax

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.10

Return last digit of the given number

Write a program that returns the last digit of the given number. Last digit is being referred to the least significant digit i.e. the digit in the ones (units) place in the given number.

The last digit should be returned as a positive number.

For example,

if the given number is 197, the last digit is

7 if the given number is -197, the last digit

is 7 **For example:**

Input	Result
123	3

Program:

```
a=int(input())
print(abs(a)%10)
```

Output:

	Input	Expected	Got	
✔	197	7	7	✔
✔	-197	7	7	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.1 IN / OUT

Ms. Sita, the faculty handling programming lab for you is very strict. Your seniors have told you that she will not allow you to enter the

week's lab if you have not completed atleast half the number of problems given last week. Many of you didn't understand this statement

and so they requested the good programmers from your batch to write a program to find whether a student will be allowed into a week's

lab given the number of problems given last week and the number of problems solved by the student in that week.

Input Format:

Input consists of 2 integers.

The first integer corresponds to the number of problems given and the second integer corresponds to the number of problems solved.

Output Format:

Output consists of the string “IN” or “OUT”.

Sample Input and Output:

Input

8

3

Output

OUT

For example:

Input Result

8

3

OUT

Program:

```
a=int(input())
b=int(input())
if(b>=a or b>=a/2):
print("IN")
else:
print("OUT")
```

Output:

	Input	Expected	Got	
✔	8 3	OUT	OUT	✔
✔	8 5	IN	IN	✔
✔	20 9	OUT	OUT	✔
✔	50 31	IN	IN	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.2 Second last digit

Write a program that returns the second last digit of the given number. Second last digit is being referred 10the digit in the tens place in

the given number.

For example, if the given number is 197, the second last digit is 9.

Note1 - The second last digit should be returned as a positive number. i.e. if the given number is -197, the second last digit is 9.

Note2 - If the given number is a single digit number, then the second last digit does not exist. In such cases, the program should return -1.

i.e. if the given number is 5, the second last digit should be returned as -1

For example:

Input Result

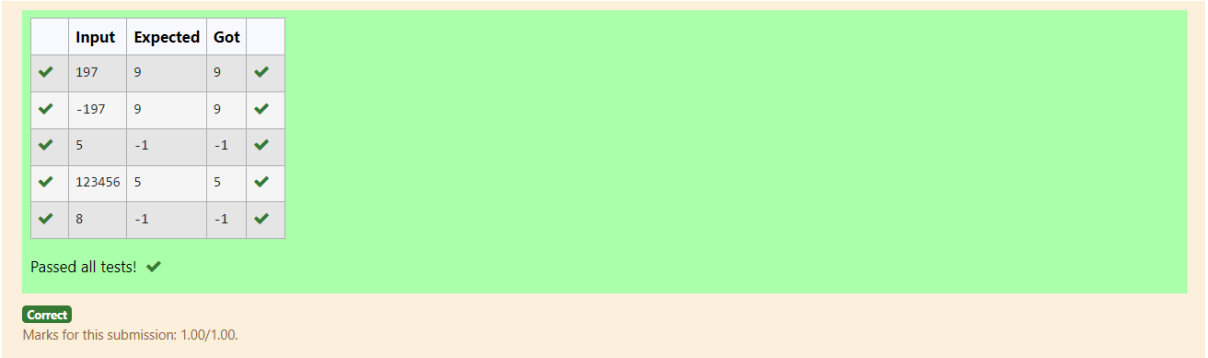
197 9

5 -1

Program:

```
a=int(input())
n=abs(a)
num=n%100
num1=num/10
if(num==n):
print("-1")
else:
print("%d"%num1)
```

Output:



	Input	Expected	Got	
✔	197	9	9	✔
✔	-197	9	9	✔
✔	5	-1	-1	✔
✔	123456	5	5	✔
✔	8	-1	-1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.3 Electricity Bill

Write a program to calculate and print the Electricity bill where the unit consumed by the user is given from test case. It prints the total

amount the customer has to pay. The charge are as follows:

Unit Charge / Unit

Upto 199 @1.20

200 and above but less than 400 @1.50

400 and above but less than 600 @1.80

600 and above @2.00

If bill exceeds Rs.400 then a surcharge of 15% will be charged and the minimum bill should be of Rs.100/-

Sample Test Cases

Test Case 1

Input

50

Output

100.00

Test Case 2

Input

300

Output

517.50

For example:

Input Result

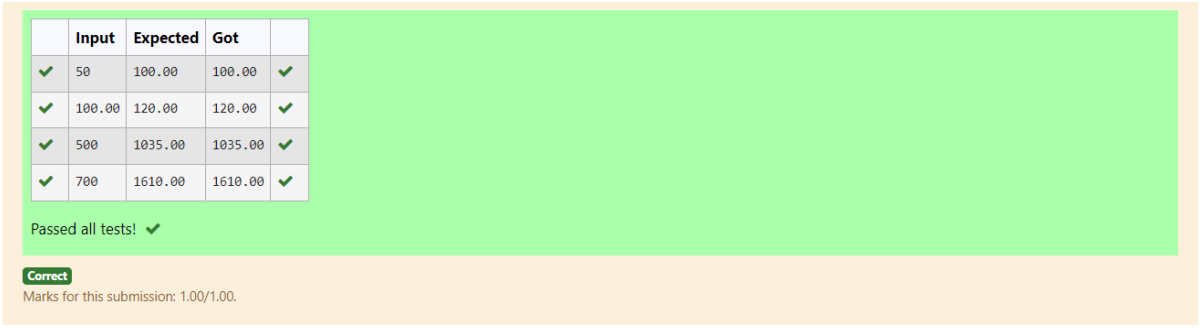
100.00 120.00

Program:

```
a=float(input())
f=a*1.20
b=a*1.50
c=a*1.80
e=0.15*c
g=c+e
d=a*2.00
```

```
h=0.15*d
i=d+h
if(a<=199):
if(f<100):
print("100.00")
else:
print("%.2f"%f)
elif(a>=200 and a<400):
print(b)
elif(a>=400 and a<600):
print("%.2f"%g)
else:
print("%.2f"%i)
```

Output:



	Input	Expected	Got	
✔	50	100.00	100.00	✔
✔	100.00	120.00	120.00	✔
✔	500	1035.00	1035.00	✔
✔	700	1610.00	1610.00	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.4 Chinese Zodiac

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

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

2010

Sample Output 1

2010 is the year of the Tiger.

Sample Input 2

2020

Sample Output 2

2020 is the year of the Rat.

Program:

```
a=int(input())
n=a%12
if(n==8):
print("%d is the year of the Dragon."%a)
```

```
elif(n==9):
print("%d is the year of the Snake."%a)
elif(n==10):
print("%d is the year of the Horse."%a)
elif(n==11):
print("%d is the year of the Sheep."%a)
elif(n==0):
print("%d is the year of the Monkey."%a)
elif(n==1):
print("%d is the year of the Rooster."%a)
elif(n==2):
print("%d is the year of the Dog."%a)
elif(n==3):
print("%d is the year of the Pig."%a)
elif(n==4):
print("%d is the year of the Rat."%a)
elif(n==5):
print("%d is the year of the Ox."%a)
elif(n==6):
print("%d is the year of the Tiger."%a)
else:
print("%d is the year of the Hare."%a)
```

Output:

	Input	Expected	Got	
✔	2010	2010 is the year of the Tiger.	2010 is the year of the Tiger.	✔
✔	2020	2020 is the year of the Rat.	2020 is the year of the Rat.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.5 Month name to days

The length of a month varies from 28 to 31 days. In this exercise you will create a program that reads the name of a month from the user as

a string. Then your program should display the number of days in that month. Display “28 or 29 days” for February so that leap years are

addressed.

Sample Input 1

February

Sample Output 1

February has 28 or 29 days in it.

Sample Input 2

March

Sample Output 2

March has 31 days in it.

Sample Input 3

April

Sample Output 3

April has 30 days in it.

For example:

Input Result

February February has 28 or 29 days in it.

Program:

```
a=input()
if(a=="January"):
print ("January has 31 days in it.")
elif(a=="February"):
print ("February has 28 or 29 days in it.")
elif(a=="March"):
print ("March has 31 days in it.")
elif(a=="April"):
print ("April has 30 days in it.")
elif(a=="May"):
print ("May has 31 days in it.")
```

```
elif(a=="June"):
print ("June has 30 days in it.")
elif(a=="July"):
print ("July has 31 days in it.")
elif(a=="August"):
print ("August has 31 days in it.")
elif(a=="September"):
print ("September has 30 days in it.")
elif(a=="October"):
print ("October has 31 days in it.")
elif(a=="November"):
print ("November has 30 days in it.")
else:
print ("December has 31 days in it.")
```

	Input	Expected	Got	
✔	February	February has 28 or 29 days in it.	February has 28 or 29 days in it.	✔
✔	March	March has 31 days in it.	March has 31 days in it.	✔
✔	April	April has 30 days in it.	April has 30 days in it.	✔
✔	May	May has 31 days in it.	May has 31 days in it.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.6 Vowel or Consonent

In this exercise you will create a program that reads a letter of the alphabet from the user. If the user enters a, e, i, o or u then your program

should display a message indicating that the entered letter is a vowel. If the user enters y then your program should display a message

indicating that sometimes y is a vowel, and sometimes y is a consonant. Otherwise your program should display a message indicating that

the letter is a consonant.

Sample Input 1

i

Sample Output 1

It's a vowel.

Sample Input 2

y

Sample Output 2

Sometimes it's a vowel... Sometimes it's a consonant.

Sample Output 3

It's a consonant.

For example:

Input Result

y Sometimes it's a vowel... Sometimes it's a consonant.

c It's a consonant.

Answer: (penalty regime: 0 %)

Program:

```
a=input()
if(a=="a"or a=="e" or a=="i" or a=="o" or a=="u"):
print("It's a vowel.")
elif(a=="y"):
print("Sometimes it's a vowel... Sometimes it's a consonant.")
else:
print("It's a consonant.")
```

Output:

	Input	Expected	Got	
✔	i	It's a vowel.	It's a vowel.	✔
✔	y	Sometimes it's a vowel... Sometimes it's a consonant.	Sometimes it's a vowel... Sometimes it's a consonant.	✔
✔	c	It's a consonant.	It's a consonant.	✔
✔	e	It's a vowel.	It's a vowel.	✔
✔	r	It's a consonant.	It's a consonant.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.7Admission Eligibility

Write a program to find the eligibility of admission for a professional course based on the following criteria:

Marks in Maths >= 65

Marks in Physics >= 55

Marks in Chemistry >= 50

Or

Total in all three subjects >= 180

Sample Test Cases

Test Case 1

Input

70

60

80

Output

The candidate is eligible

Test Case 2

Input

50

80

80

Output

The candidate is eligible

Test Case 3

Input

50

60

40

Output

The candidate is not eligible

For example:

Input Result

70

60

80

The candidate is eligible

Program:

```
mates=int(input())
phy=int(input())
chem=int(input())
total=maths+phy+chem
if(maths>=65 and phy>=50 and chem>=55):
print("The candidate is eligible")
elif(total>=180):
print("The candidate is eligible")
else:
print("The candidate is not eligible")
```

Output:

	Input	Expected	Got	
✔	70 60 80	The candidate is eligible	The candidate is eligible	✔
✔	50 80 80	The candidate is eligible	The candidate is eligible	✔
✔	50 60 40	The candidate is not eligible	The candidate is not eligible	✔
✔	20 10 25	The candidate is not eligible	The candidate is not eligible	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.8 Leap Year

Most years have 365 days. However, the time required for the Earth to orbit the Sun is actually slightly more than that. As a result, an extra

day, February 29, is included in some years to correct for this difference. Such years are referred to as leap years. The rules for determining

whether or not a year is a leap year follow:

• Any year that is divisible by 400 is a leap year.

• Of the remaining years, any year that is divisible by 100 is not a leap year.

• Of the remaining years, any year that is divisible by 4 is a leap year.

• All other years are not leap years.

Write a program that reads a year from the user and displays a message indicating whether or not it is a leap year.

Sample Input 1

1900

Sample Output 1

1900 is not a leap year.

Sample Input 2

2000

Sample Output 2

2000 is a leap year.

Program:

```
year=int(input())
if(year%400==0 and year%4==0):
print("%d is a leap year."%year)
else:
print("%d is not a leap year."%year)
```

Output:

	Input	Expected	Got	
✔	1900	1900 is not a leap year.	1900 is not a leap year.	✔
✔	2000	2000 is a leap year.	2000 is a leap year.	✔
✔	2100	2100 is not a leap year.	2100 is not a leap year.	✔
✔	2400	2400 is a leap year.	2400 is a leap year.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.9.Classifying Triangles

A triangle can be classified based on the lengths of its sides as equilateral, isosceles or scalene. All three sides of an equilateral triangle have

the same length. An isosceles triangle has two sides that are the same length, and a third side that is a different length. If all of the sides

have different lengths then the triangle is scalene.

Write a program that reads the lengths of the three sides of a triangle from the user. Then display a message that states the triangle's type.

Sample Input 1

60

60

60

Sample Output 1

That's a equilateral triangle

Sample Input 2

40

40

80

Sample Output 2

That's a isosceles triangle

Sample Input 3

50

60

70

Sample Output 3

That's a scalene triangle

For example:

Input Result

60

60

60

That's a equilateral triangle

40

40

80

That's a isosceles triangle

Program:

```
a=int(input())
b=int(input())
c=int(input())
if(a==b==c):
print("That's a equilateral triangle")
elif (a==b!=c or a!=b==c or a==c!=b):
print ("That's a isosceles triangle")
else:
print ("That's a scalene triangle")
```

Output:

	Input	Expected	Got	
✔	60 60 60	That's a equilateral triangle	That's a equilateral triangle	✔
✔	40 40 80	That's a isosceles triangle	That's a isosceles triangle	✔
✔	50 60 70	That's a scalene triangle	That's a scalene triangle	✔
✔	50 50 80	That's a isosceles triangle	That's a isosceles triangle	✔
✔	10 10 10	That's a equilateral triangle	That's a equilateral triangle	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.10 Pythagorean triple

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*3 + 4*4 = 25 = 5*5

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

3

5

4

Sample Output

yes

Sample Test Cases

Test Case 1

Input

3

5

4

Output

yes

Test Case 2

Input

5

8

2

Output

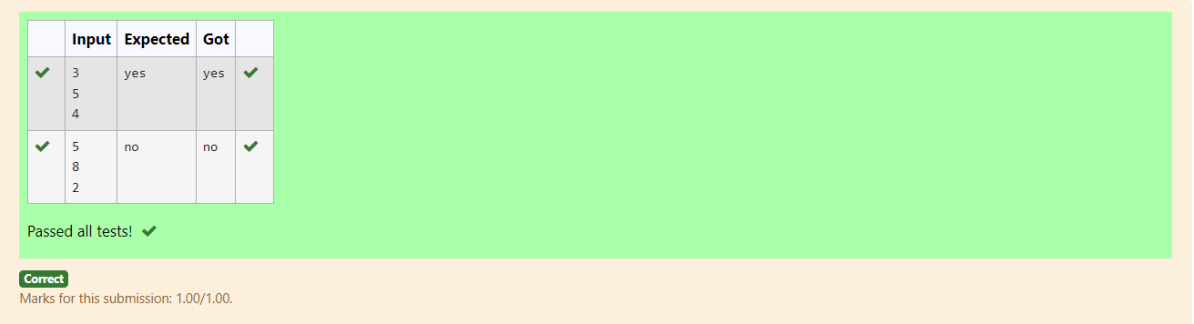
no

Program:

```
a=int(input())
b=int(input())
c=int(input())
if(a*a+b*b==c*c or b*b+c*c==a*a or a*a+c*c==b*b):
```

```
print("yes")
else:
print("no")
```

Output:



	Input	Expected	Got	
✔	3 5 4	yes	yes	✔
✔	5 8 2	no	no	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Week 4 – Algorithmic Approach:Iteration Control Structures

4.1 Factors of a number

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number)...

For example:

Input	Result	
20	1 2 4 5 10 20	

Program:

```
num=int(input())
for i in range(1,num+1):
    if num%i==0:
    print(i,end=" ")
```

Output:

	Input	Expected	Got	
✔	20	1 2 4 5 10 20	1 2 4 5 10 20	✔
✔	5	1 5	1 5	✔
✔	13	1 13	1 13	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.2 Non Repeated Digit Count

Write a program to find the count of non-repeated digits in a given number N. The number will be passed to the program as an input of type int.

Assumption: The input number will be a positive integer number >= 1 and <= 25000.
Some examples are as below.

If the given number is 292, the program should return 1 because there is only 1 non-repeated digit '9' in this number

If the given number is 1015, the program should return 2 because there are 2 non-repeated digits in this number, '0', and '5'.

If the given number is 108, the program should return 3 because there are 3 non-repeated digits in this number, '1', '0', and '8'.

If the given number is 22, the function should return 0 because there are NO non-repeated digits in this number.

For example:

Input	Result
292	1
1015	2
108	3
22	0

Program:

```
num=input()
unique_num=0
for i in num:
    n=num.count(i)
    if n==1:
        unique_num+=1
print(unique_num)
```

Output:

	Input	Expected	Got	
✔	292	1	1	✔
✔	1015	2	2	✔
✔	108	3	3	✔
✔	22	0	0	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.3 Prime Checking

Write a program that finds whether the given number N is Prime or not. If the number is prime, the program should return 2 else it must return 1.

Assumption: 2 <= N <=5000, where N is the given number.

Example1: if the given number N is 7, the method must return 2

Example2: if the given number N is 10, the method must return 1

For example:

Input	Result
7	2
10	1

Program:

```
num=int(input())
f=i
for i in range(2,num):
  if num%i==0:
     f=0
     break
if f==1:
   print("2")
else:
   print("1")
```

Output:

	Input	Expected	Got	
✔	7	2	2	✔
✔	10	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.4 Next Perfect Square

Given a number N, find the next perfect square greater than N.

Input Format:

Integer input from stdin.

Output Format:

Perfect square greater than N.

Example Input:

10

Output:

16

Program:

```
import math
n=int(input())
root=int(math.sqrt(n))
print((root+1)**2)
```

Output:

	Input	Expected	Got	
✔	10	16	16	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.5 Nth Fibonacci

Write a program to return the nth number in the fibonacci series. The value of N will be passed to the program as input.

NOTE: Fibonacci series looks like –

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . . and so on.

i.e. Fibonacci series starts with 0 and 1, and continues generating the next number as the sum of the previous two numbers.

• first Fibonacci number is 0,

• second Fibonacci number is 1,

• third Fibonacci number is 1,

• fourth Fibonacci number is 2,

• fifth Fibonacci number is 3,

• sixth Fibonacci number is 5,

• seventh Fibonacci number is 8, and so on.

For example:

Input:

7

Output:

8

Program:

```
num=int(input())
n1,n2=0,n1
for i in range(num-1):
    n1,n2=n2,n1+n2
print(n1)
```

Output:

	Input	Expected	Got	
✔	1	0	0	✔
✔	4	2	2	✔
✔	7	8	8	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.6 Disarium Number

A Number is said to be Disarium number when the sum of its digit raised to the power of their respective positions becomes equal to the number itself. Write a program to print number is Disarium or not.

Input Format:

Single Integer Input from stdin.

Output Format:

Yes or No.

Example Input:

175

Output:

Yes

Explanation

1^1 + 7^2 +5^3 = 175

Example Input:

123

Output:

No

For example:

Input	Result
175	Yes
123	No

Program:

```
num=int(input())
digit=len(str(num))
res=0
temp=num
while(temp!=0 and digit!=0):
   rem=temp%10
   temp=(temp//10)
   res+=(rem**digit)
   digit-=1
if (res==num):
   print("Yes")
else:
   print("No")
```

Output:

	Input	Expected	Got	
✔	175	Yes	Yes	✔
✔	123	No	No	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.7 Sum of Digit

Write a program to find the sum of the series 1 +11 + 111 + 1111 + . . . + n terms (n will be given as input from the user and sum will be the output)

Sample Test Cases

Test Case 1

Input

4

Output

1234

Explanation:

as input is 4, have to take 4 terms.

1 + 11 + 111 + 1111

Test Case 2

Input

6

Output

123456

For example:

Input	Result
3	123

Program:

```
n=int(input())
b=1
sum=0
for i in range(1,n+1):
    sum+=b
    b=(b*10)+1
print(sum)
```

	Input	Expected	Got	
✔	4	1234	1234	✔
✔	6	123456	123456	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.8 Unique Digit Count

Write a program to find the count of unique digits in a given number N. The number will be passed to the program as an input of type int.

Assumption: The input number will be a positive integer number >= 1 and <= 25000.

For e.g.
If the given number is 292, the program should return 2 because there are only 2 unique digits '2' and '9' in this number

If the given number is 1015, the program should return 3 because there are 3 unique digits in this number, '1', '0', and '5'.

For example:

Input	Result
292	2
1015	3

Program:

```
num=input()
unique=0
for i in range(10):
```

```
    if str(i) in num:
        unique+=1
print(unique)
```

Output:

	Input	Expected	Got	
✔	292	2	2	✔
✔	1015	3	3	✔
✔	123	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.9 Product of single digit

Given a positive integer N, check whether it can be represented as a product of single digit numbers.

Input Format:

Single Integer input.

Output Format:

Output displays Yes if condition satisfies else prints No.

Example Input:

14

Output:

Yes

Example Input:

13

Output:

No

Program:

```
a=int(input())
flag=0
for i in range(10):
    for j in range(10):
        if(i*j==a):
            flag=1
            break
if(flag==1):
    print("Yes")
else:
    print("No")
```

Output:

	Input	Expected	Got	
✔	14	Yes	Yes	✔
✔	13	No	No	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.10 Perfect Square After adding One

Given an integer N, check whether N the given number can be made a perfect square after adding 1 to it.

Input Format:

Single integer input.

Output Format:

Yes or No.

Example Input:

24

Output:

Yes

Example Input:

26

Output:

No

For example:

Input	Result
24	Yes

Program:

```
import math
n=int(input())
a=n+1
sr=int(math.sqrt(a))
if(sr*sr==a):
   print("Yes")
else:
   print("No")
```

Output:

	Input	Expected	Got	
✔	24	Yes	Yes	✔
✔	26	No	No	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Week 5 – Experiments Based on Strings and its Operations

5.1 first n chars present in s1 and s2

wo string values S1, S2 are passed as the input. The program must print first N characters present in S1 which are also present in S2.

Input Format:

The first line contains S1.
The second line contains S2.
The third line contains N.

Output Format:

The first line contains the N characters present in S1 which are also present in S2.

Boundary Conditions:

2 <= N <= 10
2 <= Length of S1, S2 <= 1000

Example Input/Output 1:

Input:

abcbde
cdefghbb
3

Output:

bcd

Note:

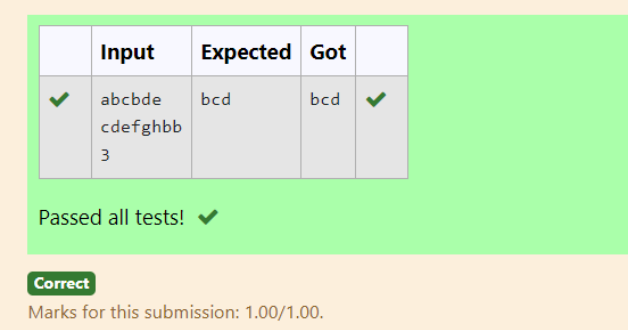
b occurs twice in common but must be printed only once.

Program:

```
def common_characters(s1, s2, n):
    result = ""
```

```
    seen = set()
    for char in s1:
        if char in s2 and char not in seen:
            result += char
            seen.add(char)
            if len(result) == n:
                break
    return result
s1 = input().strip()
s2 = input().strip()
n = int(input().strip())
print(common_characters(s1, s2, n))
```

Output:



	Input	Expected	Got	
✔	abcbde cdefghbb 3	bcd	bcd	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5.2 Count of Words whose length is L

Given a string S, which contains several words, print the count C of the words whose length is atleast L. (You can include punctuation marks like comma, full stop also as part of the word length. Space alone must be ignored)

Input Format:

The first line contains S.

The second line contains L.

Output Format:

The first line contains C

Boundary Conditions:

2 <= Length of S <= 1000

Example Input/Output 1:

Input:

During and after Kenyata’s inauguration police elsewhere in the capital, Nairobi, tried to stop the opposition from holding peaceful demonstrations.

5

Output:

13

Explanation:

The words of minimum length 5 are

During

after

Kenyatta

inauguration

police

elsewhere

capital,

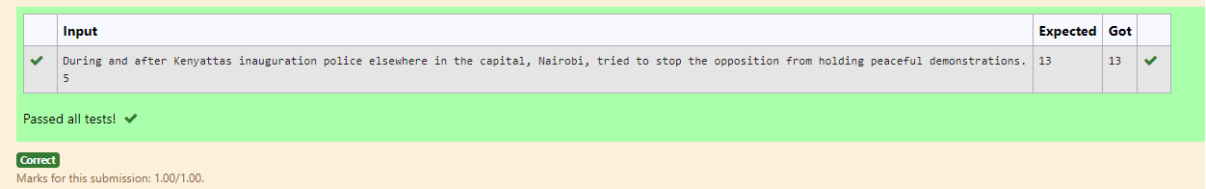
Nairobi,

tried
opposition
holding
peaceful
demonstrations.

Program:

```
def count_words_with_length_at_least_l(s, l):
    words = s.split()
    count = 0
    for word in words:
        if len(word) >= l:
            count += 1
    return count
s = input().strip()
l = int(input().strip())
print(count_words_with_length_at_least_l(s, l))
```

Output:



5.3 Longest word and its Length

Write a python to read a sentence and print its longest word and its length

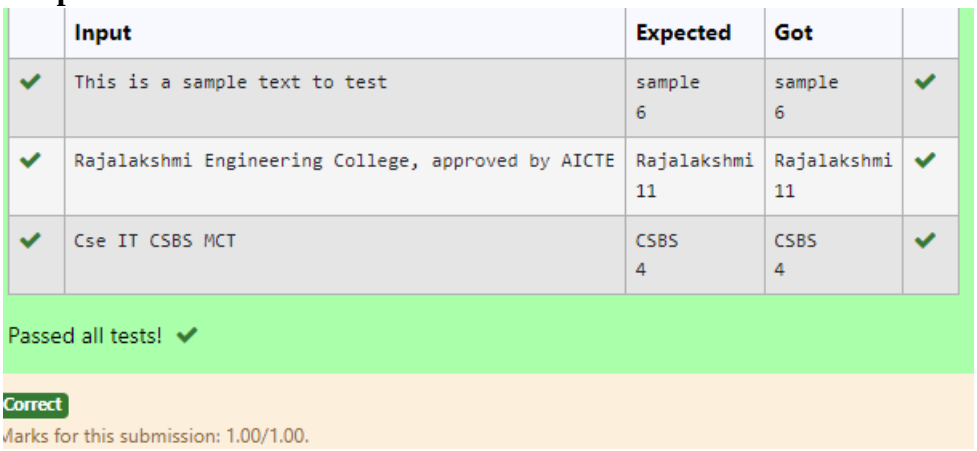
For example:

Input	Result
`This is a sample text to test`	`sample` `6`

Program:

```
def find_longest_word(sentence):
    words = sentence.split()
    longest_word = ""
    for word in words:
        if len(word) > len(longest_word):
            longest_word = word
    return longest_word, len(longest_word)
sentence = input("")
longest_word, length = find_longest_word(sentence)
print(longest_word)
print(length)
```

Output:



	Input	Expected	Got	
✔	This is a sample text to test	sample 6	sample 6	✔
✔	Rajalakshmi Engineering College, approved by AICTE	Rajalakshmi 11	Rajalakshmi 11	✔
✔	Cse IT CSBS MCT	CSBS 4	CSBS 4	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5.4 Balanced Strings

Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2. The character's position doesn't matter. If balanced display as "true", otherwise "false".

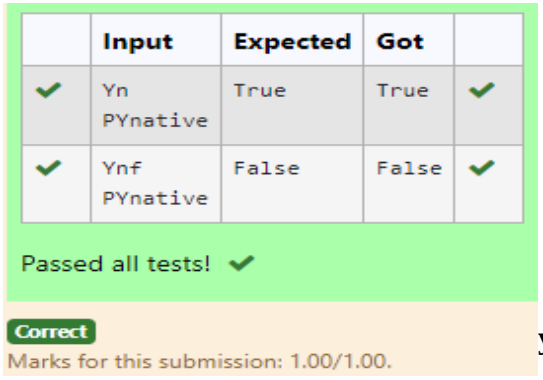
For example:

Input	Result
Yn PYnative	True

Program:

```
def are_strings_balanced(s1, s2):
    set_s1 = set(s1)
    set_s2 = set(s2)
    if set_s1.issubset(set_s2):
        return "True"
    else:
        return "False"
s1 = input().strip()
s2 = input().strip()
print(are_strings_balanced(s1, s2))
```

Output:



	Input	Expected	Got	
✔	Yn PYnative	True	True	✔
✔	Ynf PYnative	False	False	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

yword or not

Consider the below words as key words and check the given input is key word or not.

keywords: {break, case, continue, default, defer, else, for, func, goto, if, map, range, return, struct, type, var}

Input format:

Take string as an input from stdin.

Output format:

Print the word is key word or not.

Example Input:

break

Output:

break is a keyword

Example Input:

IF

Output:

IF is not a keyword

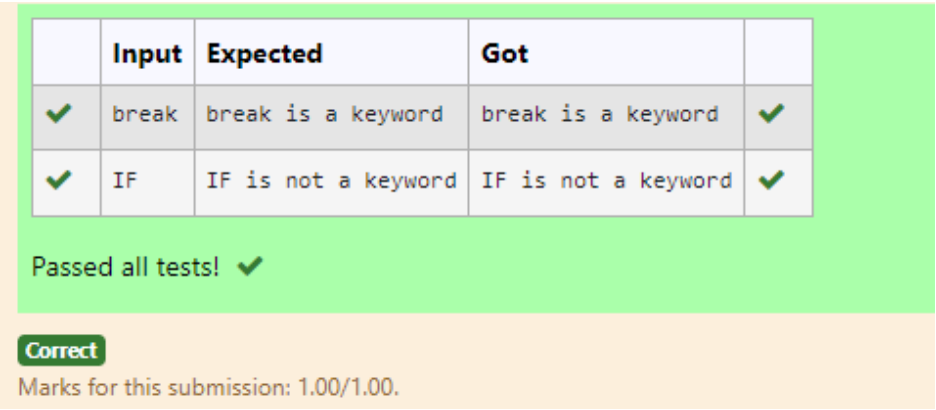
For example:

Input	Result
`break`	`break is a keyword`
`IF`	`IF is not a keyword`

Program:

```
def is_keyword(word):
    keywords = {
        "break", "case", "continue", "default", "defer", "else", "for",
        "func", "goto", "if", "map", "range", "return", "struct", "type", "var"
    }

    return word in keywords
input_word = input("")
if is_keyword(input_word):
    print("{} is a keyword".format(input_word))
else:
    print("{} is not a keyword".format(input_word))
```

Output:



	Input	Expected	Got	
✔	break	break is a keyword	break is a keyword	✔
✔	IF	IF is not a keyword	IF is not a keyword	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5.6 Count of letters,symbols,digits

Write a python program to count all letters, digits, and special symbols respectively from a given string

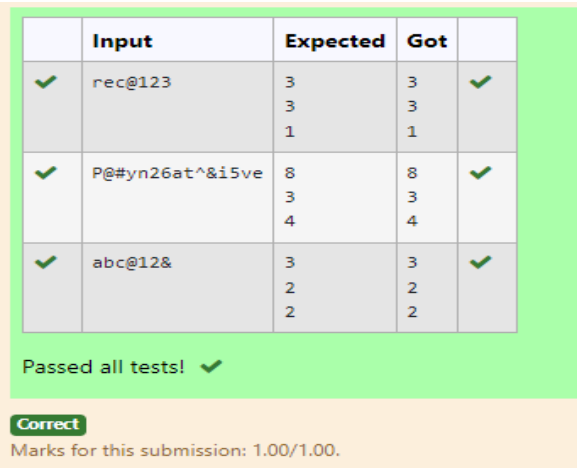
For example:

Input	Result
rec@123	3 3 1

Program:

```
def count_characters(string):
    letters = 0
    digits = 0
    special_symbols = 0
    for char in string:
        if char.isalpha():
            letters += 1
        elif char.isdigit():
            digits += 1
        else:
            special_symbols += 1
    return letters, digits, special_symbols

input_string = input("")
letters_count, digits_count, special_symbols_count = count_characters(input_string)
print(letters_count)
print(digits_count)
print(special_symbols_count)
```

Output:



	Input	Expected	Got	
✔	rec@123	3 3 1	3 3 1	✔
✔	P@#yn26at^&i5ve	8 3 4	8 3 4	✔
✔	abc@12&	3 2 2	3 2 2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5.7 Lexicographical Comparison of Strings

Robert is having 2 strings consist of uppercase & lowercase english letters. Now he wants to compare those two strings lexicographically. The letters' case does not matter, that is an uppercase letter is considered equivalent to the corresponding lowercase letter.

Input

The first line contains **T**. Then **T** test cases follow.

Each test case contains two lines contains a string. The strings' lengths range from 1 to 100 inclusive. It is guaranteed that the strings are of the same length and also consist of uppercase and lowercase Latin letters.

Output

If the first string is less than the second one, print "-1".

If the second string is less than the first one, print "1".

If the strings are equal, print "0".

Note that the letters' case is not taken into consideration when the strings are compared.

Constraint

`1≤T≤50`

`String length≤100`

For example:

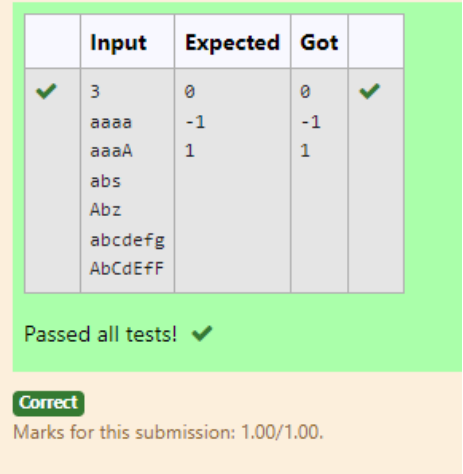
Input	Result
3 aaaa aaaA abs Abz abcdefg AbCdEfF	0 -1 1

Program:

```
def compare_strings_lexicographically(str1, str2):
    str1_lower = str1.lower()
    str2_lower = str2.lower()
    if str1_lower < str2_lower:
```

```
        return -1
    elif str1_lower > str2_lower:
        return 1
    else:
        return 0
T = int(input())
for _ in range(T):
    str1 = input()
    str2 = input()
    result = compare_strings_lexicographically(str1, str2)
    print(result)
```

Output:



	Input	Expected	Got	
✔	3 aaaa aaaA abs Abz abcdefg AbCdEfF	0 -1 1	0 -1 1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5.8 Reverse except special characters

Reverse a string without affecting special characters

Given a string **S**, containing special characters and all the alphabets, reverse the string without affecting the positions of the special characters.

Input:
A&B
Output:
B&A
Explanation: As we ignore '&' and
As we ignore '&' and then reverse, so answer is "B&A".

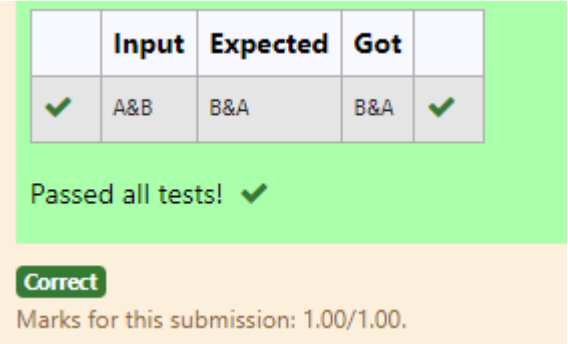
Input	Result
A&x#	x&A#

Program:

```
def reverse_string(s):
    s_list = list(s)
    n = len(s)
    special_chars = [(i, s_list[i]) for i in range(n) if not s_list[i].isalnum()]
    left, right = 0, n - 1
    while left < right:
        if s_list[left].isalnum() and s_list[right].isalnum():
            s_list[left], s_list[right] = s_list[right], s_list[left]
```

```
            left += 1
            right -= 1
        elif not s_list[left].isalnum():
            left += 1
        elif not s_list[right].isalnum():
            right -= 1
    for index, char in special_chars:
        s_list[index] = char
    reversed_string = ''.join(s_list)
    return reversed_string

input_string = input("")
print(reverse_string(input_string))
```

Output:



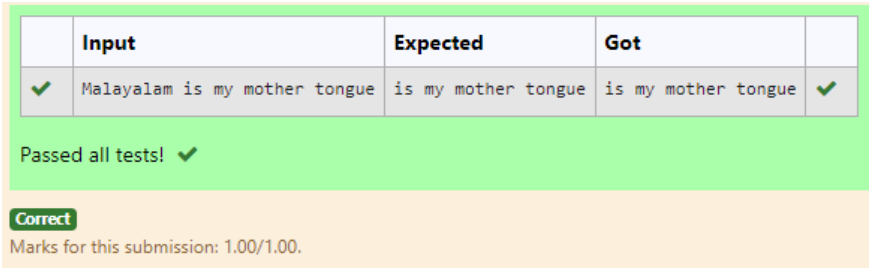
5.9 Strings without Palindrome

String should contain only the words are not palindrome.
Sample Input 1
Malayalam is my mother tongue
Sample Output 1
is my mother tongue

Program:

```
s=input()
wrds=s.split()
x=''
for word in wrds:
    word=word.lower()
    if(word!=word[::-1]):
        print(word,end=" ")
```

Output:



	Input	Expected	Got	
✔	Malayalam is my mother tongue	is my mother tongue	is my mother tongue	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5.10 Print the particulars in reverse order

Given a string S which is of the format USERNAME@DOMAIN.EXTENSION, the program must print the EXTENSION, DOMAIN, USERNAME in the reverse order.

Input Format:
The first line contains S.
Output Format:
The first line contains EXTENSION.
The second line contains DOMAIN.
The third line contains USERNAME.
Boundary Condition:
1 <= Length of S <= 100
Example Input/Output 1:
Input:
abcd@gmail.com
Output:
com
gmail
abcd

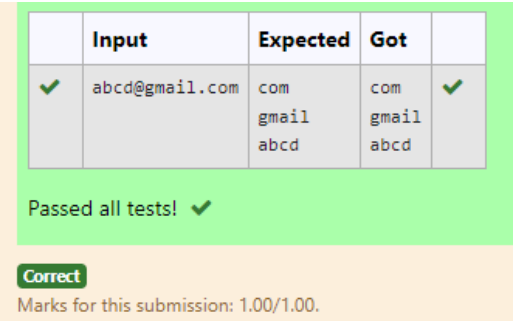
For example:

Input	Result
arvijayakumar@rajalakshmi.edu.in	edu.in rajalakshmi arvijayakumar

Program:

```
s=input()
f=s.find('@')
r=s.find('.')
print(s[r+1:])
print(s[f+1:r])
print(s[:f])
```

Output:



	Input	Expected	Got	
✔	abcd@gmail.com	com gmail abcd	com gmail abcd	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Week 6-Coding

Experiments based on lists and its operations

6.1

Given two lists A and B, and B is an anagram of A. B is an anagram of A means B is made by randomizing the order of the elements in A.

We want to find an *index mapping* P, from A to B. A mapping P[i] = j means the ith element in A appears in B at index j.

These lists A and B may contain duplicates. If there are multiple answers, output any of them.

For example, given

Input

5

12 28 46 32 50

50 12 32 46 28

Output

1 4 3 2 0

Explanation

A = [12, 28, 46, 32, 50]

B = [50, 12, 32, 46, 28]

We should return

[1, 4, 3, 2, 0]

as P [0] = 1 because the 0th element of A appears at B [1], and P [1] = 4 because the 1st element of A appears at B [4], and so on.

Note:

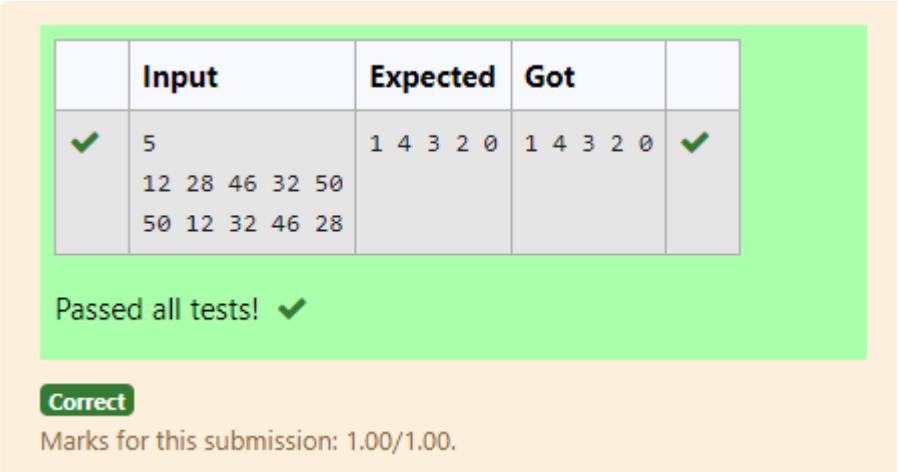
1. A, B have equal lengths in range [1, 100].

2. A[i], B[i]

Program:

```
a=input ()
b=input ()
c=input ()
l1=b.split(' ')
l2=c.split(' ')
for i in l1:
    print (l2. index (i),end=' ')
```

Output:



	Input	Expected	Got	
✔	5 12 28 46 32 50 50 12 32 46 28	1 4 3 2 0	1 4 3 2 0	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.2:

Program Description:

Program to print all the distinct elements in an array. Distinct elements are nothing but the unique (non-duplicate) elements present in the given array.

Input Format:

First line takes an Integer input from stdin which is array length n.

Second line take n Integers which is inputs of array.

Output Format:

Print the Distinct Elements in Array in single line which is space Separated

Example Input:

5

1

2

2

3

4

Output:

1 2 3 4

Example Input:

6

1

1

2

2

3

3

Output:

1 2 3

For example:

Input	Result
5 1 2 2 3 4	1 2 3 4
6 1	1 2 3

Input	Result
1 2 2 3 3	

Program:

```
a=set()
b=int(input())
for i in range(b):
    a.add(int(input()))
b=sorted(list(a))
for i in b:
    print (i,end=' ')
```

Output:

	Input	Expected	Got	
✔	5 1 2 2 3 4	1 2 3 4	1 2 3 4	✔
✔	6 1 1 2 2 3 3	1 2 3	1 2 3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.3:

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

Constraints

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

$1 \leq p \leq 10^{9}$

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

10

3

Sample Output 0

5

Explanation 0

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

Sample Case 1

Sample Input 1

10

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

1

1

Sample Output 2

1

Explanation 2

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

For example:

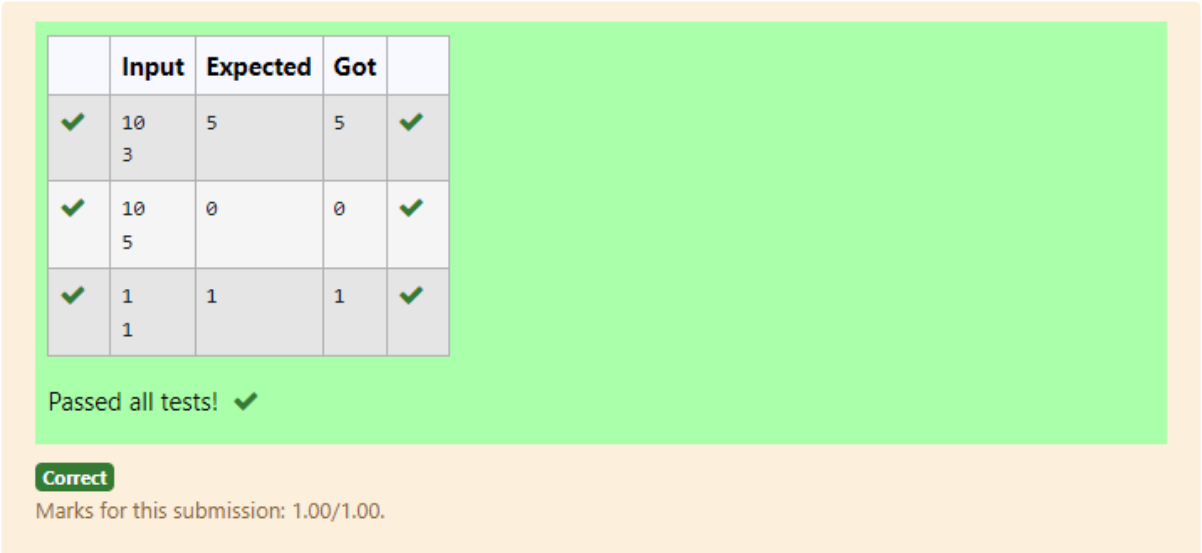
Input	Result
10 3	5
10 5	0
1 1	1

Program:

```
def find_pth_factor(n, p):
    factors = []
    for i in range(1, int(n**0.5) + 1):
        if n % i == 0:
            factors.append(i)
```

```
            if i != n // i:
                factors.append(n // i)
    factors.sort()
    if p > len(factors):
        return 0
    else:
        return factors[p-1]
n=int(input(""))
p=int(input(""))
print(find_pth_factor(n,p))
```

Output:



	Input	Expected	Got	
✔	10 3	5	5	✔
✔	10 5	0	0	✔
✔	1 1	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.4

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 != 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

Input

1

3

1

3

5

4

Output:

1

Input

1

3

1

3

5

99

Output

0

For example:

Input	Result
1 3 1 3 5 4	1
1 3 1 3 5 99	0

Program:

```
a=int(input())
for _ in range(a):
    l=[]
    s=0
    n = int(input())
    for _ in range(n):
        l.append(int(input()))
    k=int(input())
    for i in range(n):
        for j in range(i+1,n):
            if l[j]-l[i]==k and i!=j:
                s=1
                break
        if(s):
            break
    print(s)
```

Output:

	Input	Expected	Got	
✔	1 3 1 3 5 4	1	1	✔
✔	1 3 1 3 5 99	0	0	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.5:

Consider a program to insert an element / item in the sorted array. Complete the logic by filling up required code in editable section. Consider an array of size 10. The eleventh item is the data is to be inserted.

Sample Test Cases

Test Case 1

Input

1
3
4
5
6
7
8
9
10
11
2

Output

ITEM to be inserted:2

After insertion array is:
1
2
3
4
5
6
7
8
9
10
11

Test Case 2

Input

11
22
33
55
66
77
88
99
110
120
44

Output

ITEM to be inserted:44
After insertion array is:
11
22
33
44
55
66
77
88
99
110
120

Program:

```
l=[]
for i in range(10):
    l.append(int(input()))
c=int(input())
l.append(c)
l=sorted(l)
print(f"ITEM to be inserted:{c}\nAfter insertion array is:")
for i in l:
    print(i)
```

Output:

	Input	Expected	Got	
✔	1 3 4 5 6 7 8 9 10 11 2	ITEM to be inserted:2 After insertion array is: 1 2 3 4 5 6 7 8 9 10 11	ITEM to be inserted:2 After insertion array is: 1 2 3 4 5 6 7 8 9 10 11	✔
✔	11 22 33 55 66 77 88 99 110 120 44	ITEM to be inserted:44 After insertion array is: 11 22 33 44 55 66 77 88 99 110 120	ITEM to be inserted:44 After insertion array is: 11 22 33 44 55 66 77 88 99 110 120	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.6

Write a Python program to Zip two given lists of lists.

Input:

m : row size

n: column size

list1 and list 2 : Two lists

Output

Zipped List : List which combined both list1 and list2

Sample test case

Sample input

2

2

1

3

5

7

2

4

6

8

Sample Output

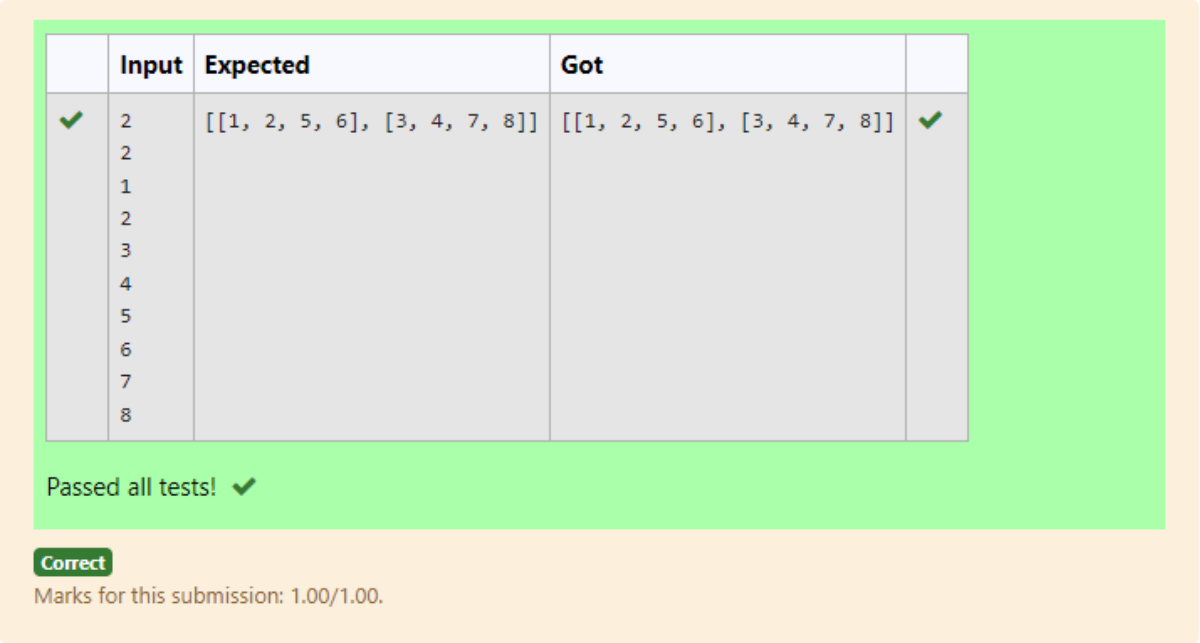
[[1, 3, 2, 4], [5, 7, 6, 8]]

Program

```
m=int(input())
n=int(input())
l1=[]
l2=[]
l=[]
for i in range(m):
    temp=[]
    for i in range(n):
        temp.append(int(input()))
    l1.append(temp)
for i in range(m):
    temp=[]
    for i in range(n):
        temp.append(int(input ()))
    l2.append(temp)
for i in range(m):
```

```
    l.append(l1[i]+l2[i])
print(l)
```

Output:



	Input	Expected	Got	
✔	2 2 1 2 3 4 5 6 7 8	[[1, 2, 5, 6], [3, 4, 7, 8]]	[[1, 2, 5, 6], [3, 4, 7, 8]]	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.7:

Write a program to print all the locations at which a particular element (taken as input) is found in a list and also print the total number of times it occurs in the list. The location starts from 1.

For example, if there are 4 elements in the array:

5
6
5
7

If the element to search is 5 then the output will be:

5 is present at location 1
5 is present at location 3
5 is present 2 times in the array.

Sample Test Cases

Test Case 1

Input

4
5
6
5
7
5

Output

5 is present at location 1.
5 is present at location 3.
5 is present 2 times in the array.

Test Case 2

Input

5
67
80
45
97
100
50

Output

50 is not present in the array.

Program:

```
def find_element_locations(arr, target):
    locations = []
    count = 0
    for i in range(len(arr)):
        if arr[i] == target:
            locations.append(i + 1)
            count += 1
```

```
    return locations, count

# Input the number of elements in the list
n = int(input(""))

# Input the elements of the list
arr = []
for i in range(n):
    arr.append(int(input("")))

# Input the target element to search for
target = int(input(""))

locations, count = find_element_locations(arr, target)

if count > 0:
    for loc in locations:
        print(f"{target} is present at location {loc}.")
    print(f"{target} is present {count} times in the array.")
else:
    print(f"{target} is not present in the array.")
```

Output:

	Input	Expected	Got	
✔	4 5 6 5 7 5	5 is present at location 1. 5 is present at location 3. 5 is present 2 times in the array.	5 is present at location 1. 5 is present at location 3. 5 is present 2 times in the array.	✔
✔	5 67 80 45 97 100 50	50 is not present in the array.	50 is not present in the array.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.8:

Write a Python program to check if a given list is strictly increasing or not. Moreover, if removing only one element from the list results in a strictly increasing list, we still consider the list true

Input:

n : Number of elements

List1: List of values

Output

Print "True" if list is strictly increasing or decreasing else print "False"

Sample Test Case

Input

7

1

2

3

0

4

5

6

Output

True

Program:

```
l=[]
temp=l
a=int(input())
for i in range(a):
    s=int(input())
    l.append(s)
l1=sorted(l)
k=l1[0]
for i in l:
    if(i-1!=k):
        l.remove(i)
        l1.remove(i)
        break
    else:
        k=i
print(l==l1 or l==l1[::-1])
```

output:

	Input	Expected	Got	
✔	7 1 2 3 0 4 5 6	True	True	✔
✔	4 2 1 0 -1	True	True	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.9:

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

7

1

2

3

3

4

5

6

2

1

6

Output:

1 6

For example:

Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57
1 7 1 2 3 3 4 5 6 2 1 6	1 6

Program:

```
a=int(input())
for _ in range(a):
    l1=[]
    l2=[]
    n=int(input())
    for i in range(n):
        l1.append(int(input()))
    m=int(input())
    for j in range(m):
        l2.append(int(input()))
    i=0
    j=0
    while(i! =n and j!=m):
        if(l1[i]==l2[j]):
            print(l1[i],end=' ')
            i+=1
            j+=1
        elif(l1[i]<l2[j]):
            i+=1
        else:
            j+=1
```

output:

	Input	Expected	Got	
✔	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✔
✔	1 7 1 2 3 3 4 5 6 2 1 6	1 6	1 6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6.10:

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.

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.

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

4

1

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.

· Using zero based indexing, arr[2]=3 is the pivot between the two subarrays.

· The index of the pivot is 2.

Sample Case 1

Sample Input 1

3

1

2

1

Sample Output 1

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.

For example:

Input	Result
4 1	2

Input	Result
2 3 3	
3 1 2 1	1

Program:

```
a=int(input())
l=[]
for i in range(a):
    l.append(int(input()))
c=sum(l)//2
q=0
for j in l:
    q+=j
    if q >=c:
        q=j
        break;
print(l.index(q))
```

output:

	Input	Expected	Got	
✔	4 1 2 3 3	2	2	✔
✔	3 1 2 1	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Week 7-Coding

Experiments based on Tuples,Sets and its operations

7.1

Given a tuple and a positive integer k, the task is to find the count of distinct pairs in the tuple whose sum is equal to **K**.

Examples:

Input: t = (5, 6, 5, 7, 7, 8), K = 13
Output: 2
Explanation:
Pairs with sum K(= 13) are {(5, 8), (6, 7), (6, 7)}.
Therefore, distinct pairs with sum K(= 13) are { (5, 8), (6, 7) }.
Therefore, the required output is 2.

For example:

Input	Result
1,2,1,2,5 3	1
1,2 0	0

Program:

```
def count_distinct_pairs_with_sum(t, K):
    num_count = {}
    distinct_pairs = set()

    for num in t:
        complement = K - num
        if complement in num_count and num_count[complement] > 0:
            pair = (min(num, complement), max(num, complement))
            distinct_pairs.add(pair)
            num_count[complement] -= 1
        if num in num_count:
            num_count[num] += 1
        else:
            num_count[num] = 1

    return len(distinct_pairs)

def main():
    # Reading tuple from input
    t = tuple(map(int, input().split(',')))
```

```
    # Reading K from input
    K = int(input())

    # Calling the function with input values
    result = count_distinct_pairs_with_sum(t, K)

    # Printing the result
    print(f"{result}")

if __name__ == "__main__":
    main()
```

Output:

	Input	Expected	Got	
✔	5,6,5,7,7,8 13	2	2	✔
✔	1,2,1,2,5 3	1	1	✔
✔	1,2 0	0	0	✔

Passed all tests! ✔

7.2:

Write a program to eliminate the common elements in the given 2 arrays and print only the non-repeating

elements and the total number of such non-repeating elements.

Input Format:

The first line contains space-separated values, denoting the size of the two arrays in integer format respectively.

The next two lines contain the space-separated integer arrays to be compared.

Sample Input:

5 4

1 2 8 6 5

2 6 8 10

Sample Output:

1 5 10

3

Sample Input:

5 5

1 2 3 4 5

1 2 3 4 5

Sample Output:

NO SUCH ELEMENTS

For example:

Input	Result
5 4 1 2 8 6 5 2 6 8 10	1 5 10 3

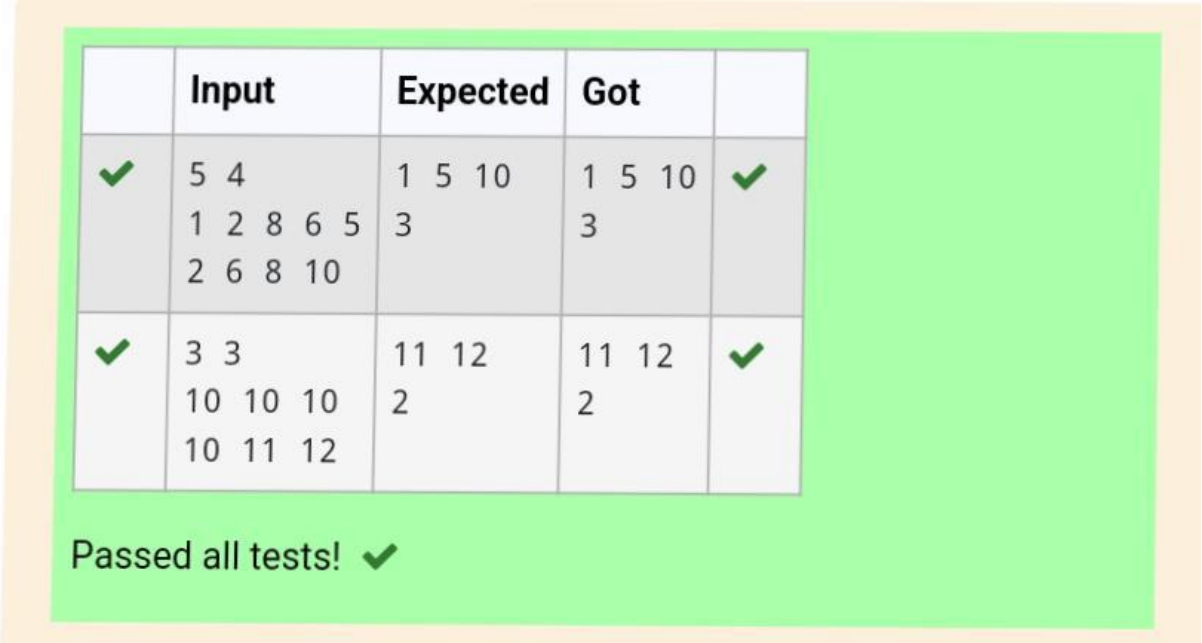
Program:

```
def find_non_repeating_elements(size1, size2, arr1, arr2):
```

```
    set1 = set(arr1)
    set2 = set(arr2)

    # Non-repeating elements in arr1
    non_repeating_arr1 = set1 - set2

    # Non-repeating elements in arr2
    non_repeating_arr2 = set2 - set1

    # Combine all non-repeating elements
    non_repeating_elements = non_repeating_arr1.union(non_repeating_arr2)

    # Sort the non-repeating elements for consistent output order
    sorted_non_repeating_elements = sorted(non_repeating_elements)

    # Print non-repeating elements
    for element in sorted_non_repeating_elements:
        print (element, end=" ")
    print ()  # Print newline

    # Print total count of non-repeating elements
    print(len(sorted_non_repeating_elements))

# Example usage with input reading
def main():
    size1, size2 = map(int, input().split())
    arr1 = list (map (int, input (). split ()))
    arr2 = list(map(int, input().split()))
```

```
    find_non_repeating_elements(size1, size2, arr1, arr2)


if __name__ == "__main__":
    main()
```

Output:



	Input	Expected	Got	
✔	5 4 1 2 8 6 5 2 6 8 10	1 5 10 3	1 5 10 3	✔
✔	3 3 10 10 10 10 11 12	11 12 2	11 12 2	✔

Passed all tests! ✔

7.3:

Given an array of integers `nums` containing `n + 1` integers where each integer is in the range `[1, n]` inclusive.There is only **one repeated number** in `nums`, return *this repeated number*. Solve the problem using set.

Example 1:

```
Input: nums = [1,3,4,2,2]
Output: 2
```

Example 2:

```
Input: nums = [3,1,3,4,2]
Output: 3
```

For example:

Input	Result
1 3 4 4 2	4

Program:

```
def find_duplicate(nums):
    seen = set()

    for num in nums:
        if num in seen:
            return num
        seen.add(num)

# Example usage with user input:
def main():
    nums = list (map(int, input().split()))
    result = find_duplicate(nums)
    print(f"{result}")

if __name__ == "__main__":
    main()
```

Output:

	Input	Expected	Got	
✔	1 3 4 4 2	4	4	✔
✔	1 2 2 3 4 5 6 7	2	2	✔

Passed all tests! ✔

7.4

There is a malfunctioning keyboard where some letter keys do not work. All other keys on the keyboard work properly.

Given a string text of words separated by a single space (no leading or trailing spaces) and a string brokenLetters of all distinct letter keys that are broken, return the number of words in text you can fully type using this keyboard.

Example 1:

Input: text = "hello world", brokenLetters = "ad"

Output:

1

Explanation: We cannot type "world" because the 'd' key is broken.

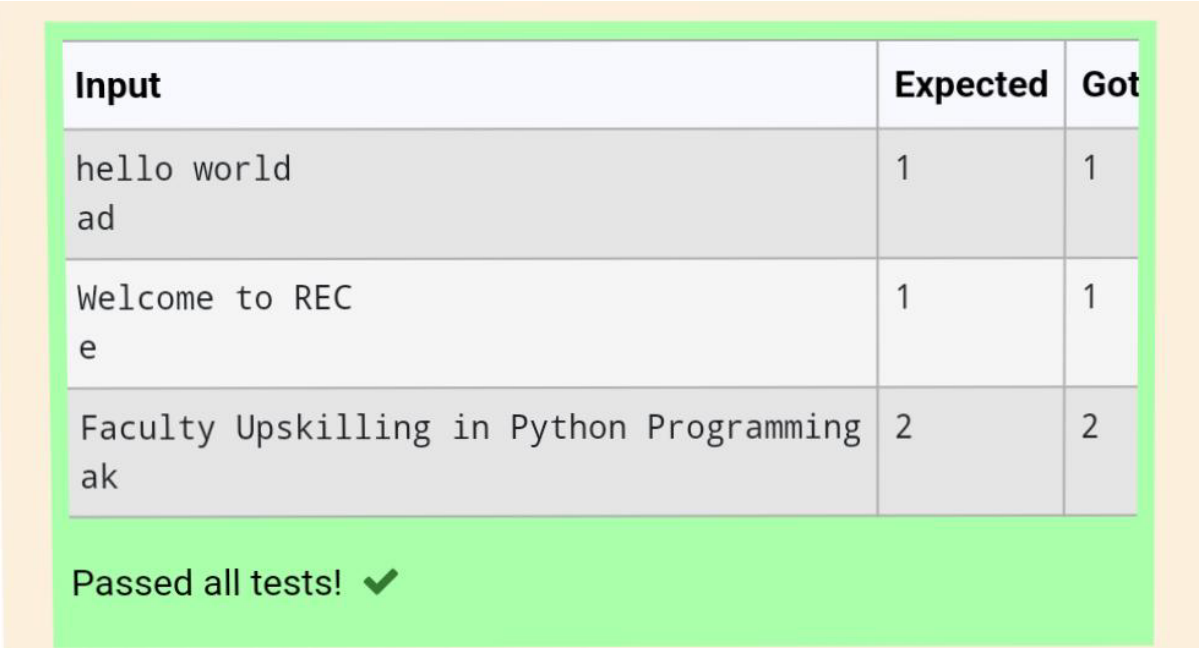
For example:

Input	Result
hello world ad	1
Faculty Upskilling in Python Programming ak	2

Program:

```
def can_type_words(text, brokenLetters):
    words = text.split(' ')  # Split by single space to preserve words correctly
    broken_set = set(brokenLetters)
```

```
    count = 0

    for word in words:
        is_broken = False
        for char in word:
            if char.lower() in broken_set:
                is_broken = True
                break
        if not is_broken:
            count += 1

    return count

# Example usage:
text = input()
brokenLetters = input()
print(can_type_words(text, brokenLetters))
```



Input	Expected	Got
hello world ad	1	1
Welcome to REC e	1	1
Faculty Upskilling in Python Programming ak	2	2

Passed all tests! ✔

7.5:

Coders here is a simple task for you, Given string str. Your task is to check whether it is a binary string or not by using python set.

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

For example:

Input	Result
01010101010	Yes
010101 10101	No

Program:

```
is_binary_string(s):
    # Create a set from the string characters
    s_set = set(s)

    # Check if the set contains only '0' and '1'
    if s_set == {'0', '1'} or s_set == {'0'} or s_set == {'1'}:
        return "Yes"
    else:
        return "No"

# Examples:
str1 =input()
```

print(is_binary_string(str1)) # Output: Yes

#print(is_binary_string(str2)) # Output: No

Output:

	Input	Expected	Got	
✔	01010101010	Yes	Yes	✔
✔	REC123	No	No	✔
✔	010101 10101	No	No	✔

Passed all tests! ✔

Experiments based on Dictionary and its operations.

8.1

Create a student dictionary for n students with the student name as key and their test mark assignment mark and lab mark as values. Do the following computations and display the result.

1.Identify the student with the highest average score

2.Identify the student who as the highest Assignment marks

3.Identify the student with the Lowest lab marks

4.Identify the student with the lowest average score

Note:

If more than one student has the same score display all the student names

Sample input:

4

James 67 89 56

Lalith 89 45 45

Ram 89 89 89

Sita 70 70 70

Sample Output:

Ram

James Ram

Lalith

Lalith

For example:

Input	Result
4 James 67 89 56 Lalith 89 45 45 Ram 89 89 89 Sita 70 70 70	Ram James Ram Lalith Lalith

Program:

```
Def main():
    # Input number of students
    N = int(input().strip())

    # Initialize an empty dictionary to store student data
    Student_dict = {}
```

```
    # Input student data
    For _ in range(n):
        Data = input().strip().split()
        Name = data[0]
        Test_mark = int(data[1])
        Assignment_mark = int(data[2])
        Lab_mark = int(data[3])
        Student_dict[name] = (test_mark, assignment_mark, lab_mark)

    # Compute average scores
    Average_scores = {name: sum(marks)/3 for name, marks in
student_dict.items()}

    # Identify the student(s) with the highest average score
    Highest_avg_score = max(average_scores.values())
    Highest_avg_students = sorted([name for name, avg in
average_scores.items() if avg == highest_avg_score])

    # Identify the student(s) with the highest assignment marks
    Highest_assignment_mark = max(student_dict. values(), key=lambda x:
x[1])[1]
    Highest_assignment_students = sorted([name for name, marks in
student_dict.items() if marks[1] == highest_assignment_mark])

    # Identify the student(s) with the lowest lab marks
    Lowest_lab_mark = min(student_dict.values(), key=lambda x: x[2])[2]
    Lowest_lab_students = sorted ([name for name, marks in student_dict.
items() if marks [2] == lowest_lab_mark])
```

```
    # Identify the student(s) with the lowest average score
    Lowest_avg_score = min(average_scores.values())
    Lowest_avg_students = sorted([name for name, avg in
average_scores.items() if avg == lowest_avg_score])

    # Print results
    Print(“ “.join(highest_avg_students))
    Print(“ “.join(highest_assignment_students))
    Print(“ “.join(lowest_lab_students))
    Print(“ “.join(lowest_avg_students))

If __name__ == "__main__":
    main()
```

Output:

	Input	Expected	Got	
✔	4 James 67 89 56 Lalith 89 45 45 Ram 89 89 89 Sita 70 70 70	Ram James Ram Lalith Lalith	Ram James Ram Lalith Lalith	✔
✔	3 Raja 95 67 90 Aarav 89 90 90 Shadhana 95 95 91	Shadhana Shadhana Aarav Raja Raja	Shadhana Shadhana Aarav Raja Raja	✔

Passed all tests! ✔

8.2:

A sentence is a string of single-space separated words where each word consists only of lowercase letters. A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences s1 and s2, return a list of all the uncommon words. You may return the answer in any order.

Example 1:

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

Constraints:

1 <= s1.length, s2.length <= 200

s1 and s2 consist of lowercase English letters and spaces.

s1 and s2 do not have leading or trailing spaces.

All the words in s1 and s2 are separated by a single space.

Note:

Use dictionary to solve the problem

For example:

Input	Result
this apple is sweet this apple is sour	sweet sour

Program:

```
def uncommon_from_sentences(s1, s2):
    from collections import Counter

    # Split sentences into words
    words1 = s1.split()
    words2 = s2.split()

    # Count occurrences of each word in both sentences
    count = Counter(words1 + words2)

    # Find words that appear exactly once in total and are in either s1 or s2 but not both
    uncommon_words = [word for word in count if count[word] == 1 and (word in words1 or word in words2)]

    return uncommon_words

# Sample Inputs
s1 = input()
s2 = input()

# Find and print uncommon words
result = uncommon_from_sentences(s1, s2)
```

```
print(" ".join(result))


# Additional example
```

Output:

	Input	Expected	Got	
✔	this apple is sweet this apple is sour	sweet sour	sweet sour	✔
✔	apple apple banana	banana	banana	✔

Passed all tests! ✔

8.3:

In the game of Scrabble™, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points. The points associated with each letter are shown below:

Points Letters

1 A, E, I, L, N, O, R, S, T and U

2 D and G

3 B, C, M and P

4 F, H, V, W and Y

5 K

8 J and X

10 Q and Z

Write a program that computes and displays the Scrabble™ score for a word. Create a dictionary that maps from letters to point values. Then use the dictionary to compute the score.

A Scrabble™ board includes some squares that multiply the value of a letter or the value of an entire word. We will ignore these squares in this exercise.

Sample Input

REC

Sample Output

REC is worth 5 points.

For example:

Input	Result
REC	REC is worth 5 points.

Program:

```
# Define the points dictionary
points = {
    1: 'AEILNORSTU',
    2: 'DG',
    3: 'BCMP',
    4: 'FHVWY',
    5: 'K',
    8: 'JX',
    10: 'QZ'
}

# Create a dictionary that maps letters to their corresponding point values
letter_to_points = {}
for point_value, letters in points.items():
    for letter in letters:
        letter_to_points[letter] = point_value
```

```
# Function to calculate the Scrabble™ score of a word
def scrabble score(word):
    score = 0
    for letter in word.upper():
        score += letter_to_points.get(letter, 0)
    return score

# Sample input
word = input()

# Calculate the score
score = scrabble_score(word)

# Display the result
print(f"{word} is worth {score} points.")
```

	Input	Expected	Got
✔	GOD	GOD is worth 5 points.	GOD is worth 5 poin
✔	REC	REC is worth 5 points.	REC is worth 5 poin

Passed all tests! ✔

8.4:

Give a dictionary with value lists, sort the keys by summation of values in value list.

Input: test_dict = {‘Gfg’ : [6, 7, 4], ‘best’ : [7, 6, 5]}

Output: {‘Gfg’: 17, ‘best’: 18}

Explanation: Sorted by sum, and replaced.

Input: test_dict = {‘Gfg’ : [8,8], ‘best’ : [5,5]}

Output: {‘best’: 10, ‘Gfg’: 16}

Explanation: Sorted by sum, and replaced.

Sample Input:

2

Gfg 6 7 4

Best 7 6 5

Sample Output

Gfg 17

Best 18

For example:

Input	Result
2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18

Program:

```
# Function to read input, process the dictionary, and produce the output
def process_dictionary():
    # Read the number of entries
    num_entries = int(input())

    # Initialize the dictionary
    test_dict = {}

    # Read the entries
    for _ in range(num_entries):
        line = input().split()
        key = line[0]
        values = list(map(int, line[1:]))
```

```
        test_dict[key] = values

    # Create a new dictionary with the sums of the values
    summed_dict = {key: sum(values) for key, values in test_dict.items()}

    # Sort the dictionary by the sums
    sorted_dict = dict(sorted(summed_dict.items(), key=lambda item: item[1]))

    # Produce the output
    for key, value in sorted_dict.items():
        print(f"{key} {value}")

# Call the function to process the dictionary
process_dictionary()
```

output:

	Input	Expected	Got	
✔	2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18	Gfg 17 Best 18	✔
✔	2 Gfg 6 6 Best 5 5	Best 10 Gfg 12	Best 10 Gfg 12	✔

Passed all tests! ✔

8.5:

Given an array of names of candidates in an election. A candidate name in the array represents a vote cast to the candidate. Print the name of candidates received Max vote. If there is tie, print a lexicographically smaller name.

Examples:

Input: votes[] = {"john", "johnny", "jackie",
"johnny", "john", "jackie",
"jamie", "jamie", "john",
"johnny", "jamie", "johnny",
"john"};

Output: John

We have four Candidates with name as 'John', 'Johnny', 'jamie', 'jackie'. The candidates John and Johny get maximum votes. Since John is alphabetically smaller, we print it. Use dictionary to solve the above problem

Sample Input:

10
John
John
Johny
Jamie
Jamie
Johny
Jack
Johny
Johny
Jackie

Sample Output:

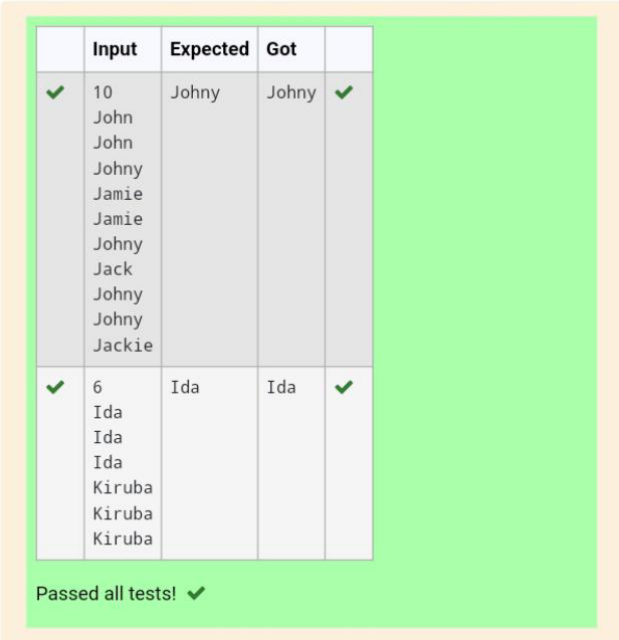
Johny

Program:

```
def find_winner(votes):
    # Dictionary to store count of each candidate
    count = {}

    # Counting votes
    for candidate in votes:
        if candidate in count:
            count[candidate] += 1
        else:
            count[candidate] = 1

    # Finding the candidate with max votes
    max_votes = -1
    winner = ""

    for candidate, votes in count.items():
        if votes > max_votes:
            max_votes = votes
            winner = candidate
        elif votes == max_votes:
            # Resolve tie by lexicographical order
            if candidate < winner:
                winner = candidate

    return winner

if __name__ == "__main__":
```

```
n = int(input())
votes = []
for _ in range(n):
    vote = input().strip()
    votes.append(vote)

winner = find_winner(votes)
print( winner)
```

output:



	Input	Expected	Got	
✔	10 John John Johny Jamie Jamie Johny Jack Johny Johny Jackie	Johny	Johny	✔
✔	6 Ida Ida Ida Kiruba Kiruba Kiruba	Ida	Ida	✔

Passed all tests! ✔

Week 9-Coding

Functions: Built-in functions, User-defined functions, Recursive functions

9.1 Program Description:

An abundant number is a number for which the sum of its proper divisors is greater than the number itself. Proper divisors of the number are those that are strictly lesser than the number.

Input Format:

Take input an integer from stdin

Output Format:

Return Yes if given number is Abundant. Otherwise, print No

Example input:

12

Output:

Yes

Explanation:

The proper divisors of 12 are: 1, 2, 3, 4, 6, whose sum is 1 + 2 + 3 + 4 + 6 = 16. Since sum of proper divisors is greater than the given number, 12 is an abundant number.

Example input:

13

Output:

No

Explanation:

The proper divisors of 13 is: 1, whose sum is 1. Since sum of proper divisors is not greater than the given number, 13 is not an abundant number.

For example:

Test	Result
`print(abundant(12))`	`Yes`
`print(abundant(13))`	`No`

Program:

```
def abundant(n):
    divisor_sum = 0
    for i in range(1, n):
        if n % i == 0:
            divisor_sum += i
    if divisor_sum > n:
        return "Yes"
    else:
        return "No"
```

Output:

	Test	Expected	Got	
✔	print(abundant(12))	Yes	Yes	✔
✔	print(abundant(13))	No	No	✔

Passed all tests! ✔

9.2 Program Description:

Write a code to check whether product of digits at even places is divisible by sum of digits at odd place of a positive integer.

Input Format:

Take an input integer from stdin.

Output Format:

Print TRUE or FALSE.

Example Input:

1256

Output:

TRUE

Example Input:

1595

Output:

FALSE

For example:

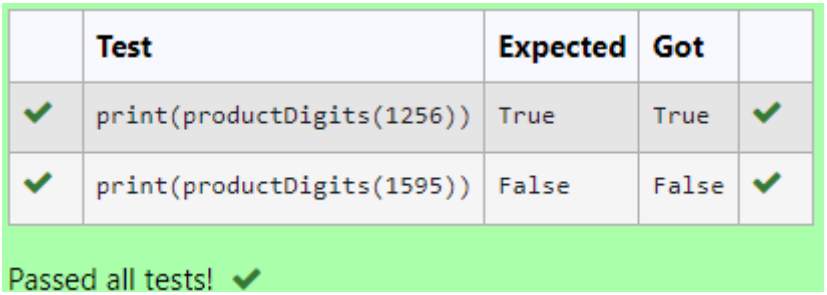
Test	Result
print(productDigits(1256))	True
print(productDigits(1595))	False

Program:

```
def productDigits(n):
 digits = [int(d) for d in str(n)]
 even_product = 1
```

```
odd_sum = 0
for i in range(len(digits)):
  if (i + 1) % 2 == 0:
    even_product *= digits[i]
  else:
     odd_sum += digits[i]
if(even_product % odd_sum == 0):
    return True
if(even_product % odd_sum !=0):
    return False
```

Output:



	Test	Expected	Got	
✔	print(productDigits(1256))	True	True	✔
✔	print(productDigits(1595))	False	False	✔

Passed all tests! ✔

9.3 Program Description:

A number is considered to be ugly if its only prime factors are 2, 3 or 5.

[1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, …] is the sequence of ugly numbers.

Task:

complete the function which takes a number n as input and checks if it's an ugly number.

return ugly if it is ugly, else return not ugly

Hint:

An ugly number U can be expressed as: U = 2^a * 3^b * 5^c, where a, b and c are nonnegative integers.

For example:

Test	Result
`print(checkUgly(6))`	`ugly`
`print(checkUgly(21))`	`not ugly`

Program:

```
def checkUgly(n):
    if n <= 0:
        return "not ugly"
    while n % 2 == 0:
        n /= 2
    while n % 3 == 0:
        n /= 3
    while n % 5 == 0:
        n /= 5
    if n == 1:
        return "ugly"
    else:
        return "not ugly"
```

Output:

	Test	Expected	Got	
✔	`print(checkUgly(6))`	`ugly`	`ugly`	✔
✔	`print(checkUgly(21))`	`not ugly`	`not ugly`	✔

Passed all tests! ✔

9.4 Program Description:

Given a number with maximum of 100 digits as input, find the difference between the sum of odd and even position digits.

Input Format:

Take a number in the form of String from stdin.

Output Format:

Print the difference between sum of even and odd digits

Example input:

1453

Output:

1

Explanation:

Here, sum of even digits is 4 + 3 = 7

sum of odd digits is 1 + 5 = 6.

Difference is 1.

Note that we are always taking absolute difference

Program:

```
def differenceSum(n):
    num_str = str(n)
    even_sum = 0
    odd_sum = 0
    for i in range(len(num_str)):
        digit = int(num_str[i])
        if (i + 1) % 2 == 0:
            even_sum += digit
        else:
            odd_sum += digit
    return abs(even_sum - odd_sum)
```

Output:

	Test	Expected	Got	
✔	print(differenceSum(1453))	1	1	✔

Passed all tests! ✔

9.5 Program Description:

Complete function to implement coin change making problem i.e. finding the minimum number of coins of certain denominations that add up to given amount of money.
The only available coins are of values 1, 2, 3, 4

Input Format:
Integer input from stdin.
Output Format:
return the minimum number of coins required to meet the given target.
Example Input:
16
Output:
4
Explanation:
We need only 4 coins of value 4 each
Example Input:
25
Output:
7
Explanation:
We need 6 coins of 4 value, and 1 coin of 1 value

Program:

```
def coinChange(n):
    coins = [1, 2, 3, 4]
    min_coins = [float('inf')] * (n + 1)
    min_coins[0] = 0
    for amount in range(1, n + 1):
        for coin in coins:
            if amount - coin >= 0:
                min_coins[amount] = min(min_coins[amount], min_coins[amount - coin] + 1)
    return min_coins[n]
```

Output:

	Test	Expected	Got	
✔	print(coinChange(16))	4	4	✔

Passed all tests! ✔

Week 10-Coding

Searching Techniques: Linear and Binary

10.1 Program Description:

Write a Python program for binary search.

For example:

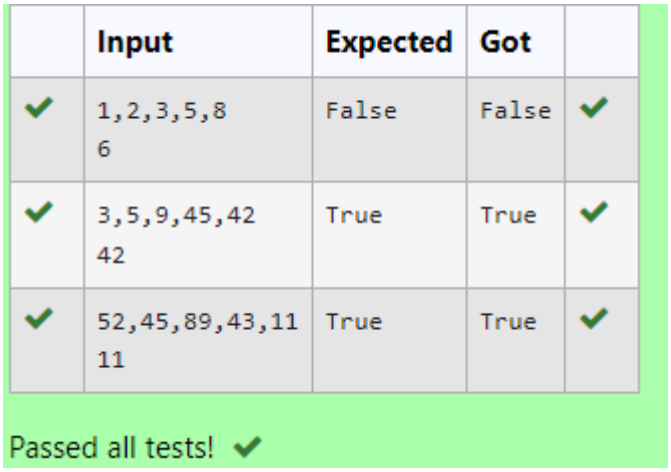
Input	Result
1,2,3,5,8 6	False
3,5,9,45,42 42	True

Program:

```
a=list(map(int, input().split(',')))
b=int(input())
a.sort()
c=0
d=len(a)-1
e=False
while c<=d:
    f=(c + d)//2
    if a[f]==b:
        e=True
        break
    elif a[f]<b:
```

```
        c=f+1
    else:
        d=f-1
if e:
    print("True")
else:
    print("False")
```

Output:



	Input	Expected	Got	
✔	1,2,3,5,8 6	False	False	✔
✔	3,5,9,45,42 42	True	True	✔
✔	52,45,89,43,11 11	True	True	✔

Passed all tests! ✔

10.2 Program Description:

To find the frequency of numbers in a list and display in sorted order.

Constraints:

1<=n, arr[i]<=100

Input:

1 68 79 4 90 68 1 4 5

Output:

1 2

4 2

5 1

68 2

79 1

90 1

For example:

Input	Result
4 3 5 3 4 5	3 2 4 2 5 2

Program:

```
a=list(map(int, input().split()))
b={}
for c in a:
    if c in b:
        b[c]+=1
    else:
        b[c]=1
d=sorted(b.items())
for e,f in d:
    print(e, f)
```

Output:

	Input	Expected	Got	
✔	4 3 5 3 4 5	3 2 4 2 5 2	3 2 4 2 5 2	✔
✔	12 4 4 4 2 3 5	2 1 3 1 4 3 5 1 12 1	2 1 3 1 4 3 5 1 12 1	✔
✔	5 4 5 4 6 5 7 3	3 1 4 2 5 3 6 1 7 1	3 1 4 2 5 3 6 1 7 1	✔

Passed all tests! ✔

10.3 Program Description:

Write a Python program to sort a list of elements using merge sort algorithm.

For example:

Input	Result
5 6 5 4 3 8	3 4 5 6 8

Program:

```
a=int(input())
b=list(map(int,input().split()))
stack=[(0, a)]
while stack:
    c,d=stack.pop()
    if d-c > 1:
        e=(c+d)//2
        stack.append((c,e))
        stack.append((e,d))
        b[c:d]=sorted(b[c:d])
print(*b)
```

Output:

	Input	Expected	Got	
✔	5 6 5 4 3 8	3 4 5 6 8	3 4 5 6 8	✔
✔	9 14 46 43 27 57 41 45 21 70	14 21 27 41 43 45 46 57 70	14 21 27 41 43 45 46 57 70	✔
✔	4 86 43 23 49	23 43 49 86	23 43 49 86	✔

Passed all tests! ✔

10.4 Program Description:

An list contains N numbers and you want to determine whether two of the numbers sum to a given number K. For example, if the input is 8, 4, 1, 6 and K is 10, the answer is yes (4 and 6). A number may be used twice.

Input Format

The first line contains a single integer n , the length of list

The second line contains n space-separated integers, list[i].

The third line contains integer k.

Output Format

Print Yes or No.

Sample Input

7

0 1 2 4 6 5 3

1

Sample Output

Yes

For example:

Input	Result
5 8 9 12 15 3 11	Yes
6 2 9 21 32 43 43 1 4	No

Program:

```
n = int(input())
nums = list (map (int, input(). split()))
k = int(input())
num_set = set()
for num in nums:
    if k - num in num_set:
        print("Yes")
        break
    num_set.add(num)
else:
```

```
print("No")
```

Output:

	Input	Expected	Got	
✔	5 8 9 12 15 3 11	Yes	Yes	✔
✔	6 2 9 21 32 43 43 1 4	No	No	✔
✔	6 13 42 31 4 8 9 17	Yes	Yes	✔

Passed all tests! ✔

10.5Program Description:

Given an list, find peak element in it. A peak element is an element that is greater than its neighbors.

An element a[i] is a peak element if

A[i-1] <= A[i] >=a[i+1] for middle elements. [0<i<n-1]

A[i-1] <= A[i] for last element [i=n-1]

A[i]>=A[i+1] for first element [i=0]

Input Format

The first line contains a single integer n , the length of A .
The second line contains n space-separated integers,A[i].

Output Format

Print peak numbers separated by space.

Sample Input

5

8 9 10 2 6

Sample Output

10 6

For example:

Input	Result
4 12 3 6 8	12 8

Program:

```
n = int(input())
nums = list(map(int, input().split()))
peak_elements = []
if n == 1:
    peak_elements.append(nums[0])
elif nums[0] >= nums[1]:
    peak_elements.append(nums[0])
for i in range(1, n - 1):
    if nums[i - 1] <= nums[i] >= nums[i + 1]:
        peak_elements.append(nums[i])
if nums[n - 1] >= nums[n - 2]:
    peak_elements.append(nums[n - 1])
print(*peak_elements)
```

Output:

	Input	Expected	Got	
✔	7 15 7 10 8 9 4 6	15 10 9 6	15 10 9 6	✔
✔	4 12 3 6 8	12 8	12 8	✔

Passed all tests! ✔

11.1 Write a Python program that asks the user for their age and prints a message based on the age. Ensure that the program handles cases where the input is not a valid integer.

Input Format: A single line input representing the user's age.

Output Format: Print a message based on the age or an error if the input is invalid.

For example:

Input	Result
twenty	Error: Please enter a valid age.
25	You are 25 years old.
-1	Error: Please enter a valid age.

Program:

```
try:
    age = int(input(""))
    if age < 0:
        print("Error: Please enter a valid age.")
    else:
        print(f"You are {age} years old.")
except ValueError:
    print("Error: Please enter a valid age.")
except EOFError:
    print("Error: Please enter a valid age.")
```

Output:

	Input	Expected	Got	
✔	twenty	Error: Please enter a valid age.	Error: Please enter a valid age.	✔
✔	25	You are 25 years old.	You are 25 years old.	✔
✔	-1	Error: Please enter a valid age.	Error: Please enter a valid age.	✔
✔	150	You are 150 years old.	You are 150 years old.	✔
✔		Error: Please enter a valid age.	Error: Please enter a valid age.	✔

Passed all tests! ✔

11.2 Problem Description:
Write a Python program that asks the user for their age and prints a message based on the age. Ensure that the program handles cases where the input is not a valid integer.

Input Format:

A single line input representing the user's age.

Output Format:

Print a message based on the age or an error if the input is invalid

For example:

Input	Result
25	You are 25 years old.
rec	Error: Please enter a valid age.
-5	Error: Please enter a valid age.

Program:

```
try:
    age = int(input())
    if age<=0:
        raise ValueError
except EOFError:
    print("Error: Please enter a valid age.")
except ValueError:
    print("Error: Please enter a valid age.")
else:
    print(f"You are {age} years old.")
```

Output:

	Input	Expected	Got	
✔	25	You are 25 years old.	You are 25 years old.	✔
✔	rec	Error: Please enter a valid age.	Error: Please enter a valid age.	✔
✔	!@#	Error: Please enter a valid age.	Error: Please enter a valid age.	✔

Passed all tests! ✔

11.3 Problem Description:

Develop a Python program that safely calculates the square root of a number provided by the user. Handle exceptions for negative inputs and non-numeric inputs.

Input Format:

User inputs a number.

Output Format:

Print the square root of the number or an error message if an exception occurs.

For example:

Input	Result
16	The square root of 16.0 is 4.00
-4	Error: Cannot calculate the square root of a negative number.
rec	Error: could not convert string to float

Program:

```
import math
def calculate_square_root():
    try:
        num = float(input())
        if num < 0:
            print("Error: Cannot calculate the square root of a negative number.")
        else:
            sqrt = math.sqrt(num)
            print(f"The square root of {num} is {sqrt:.2f}")
    except ValueError:
        print("Error: could not convert string to float")
calculate_square_root()
```

Output:

	Input	Expected	Got	
✔	16	The square root of 16.0 is 4.00	The square root of 16.0 is 4.00	✔
✔	0	The square root of 0.0 is 0.00	The square root of 0.0 is 0.00	✔
✔	-4	Error: Cannot calculate the square root of a negative number.	Error: Cannot calculate the square root of a negative number.	✔

Passed all tests! ✔

11.4 Problem Description:

Write a Python script that asks the user to enter a number within a specified range (e.g., 1 to 100). Handle exceptions for invalid inputs and out-of-range numbers.

Input Format:

User inputs a number.

Output Format:

Confirm the input or print an error message if it's invalid or out of range.

For example:

Input	Result
1	Valid input.
101	Error: Number out of allowed range
rec	Error: invalid literal for int()

Program:

```
try:
    lower_limit = 1
    upper_limit = 100
    num = int(input(""))
    if num < lower_limit or num > upper_limit:
        print("Error: Number out of allowed range")
    else:
        print("Valid input.")
except ValueError:
    print("Error: invalid literal for int()")
```

Output:

	Input	Expected	Got	
✔	1	Valid input.	Valid input.	✔
✔	100	Valid input.	Valid input.	✔
✔	101	Error: Number out of allowed range	Error: Number out of allowed range	✔

Passed all tests! ✔

11.5 Write a Python program that performs division and modulo operations on two numbers provided by the user. Handle division by zero and non-numeric inputs.

Input Format:

Two lines of input, each containing a number.

Output Format:

Print the result of division and modulo operation, or an error message if an exception occurs.

For example:

Input	Result
10 2	Division result: 5.0 Modulo result: 0
7 3	Division result: 2.3333333333333335 Modulo result: 1
8 0	Error: Cannot divide or modulo by zero.

Program:

```
try:
    num1 = float(input(""))
    num2 = float(input(""))
    division_result = num1 / num2
    modulo_result = num1 % num2
    print(f"Division result: {division_result:}")
    print(f"Modulo result: {modulo_result:.0f}")
except ZeroDivisionError:
    print("Error: Cannot divide or modulo by zero.")
except ValueError:
    print("Error: Non-numeric input provided.")
```

Output:

	Input	Expected	Got	
✔	10 2	Division result: 5.0 Modulo result: 0	Division result: 5.0 Modulo result: 0	✔
✔	7 3	Division result: 2.3333333333333335 Modulo result: 1	Division result: 2.3333333333333335 Modulo result: 1	✔
✔	8 0	Error: Cannot divide or modulo by zero.	Error: Cannot divide or modulo by zero.	✔
✔	abc 5	Error: Non-numeric input provided.	Error: Non-numeric input provided.	✔

Passed all tests! ✔

12.1 Background:

Rose manages a personal library with a diverse collection of books. To streamline her library management, she needs a program that can categorize books based on their genres, making it easier to find and organize her collection.

Problem Statement:

Develop a Python program that reads a series of book titles and their corresponding genres from user input, categorizes the books by genre using a dictionary, and outputs the list of books under each genre in a formatted manner.

Input Format:

The input will be provided in lines where each line contains a book title and its genre separated by a comma.

Input terminates with a blank line.

Output Format:

For each genre, output the genre name followed by a colon and a list of book titles in that genre, separated by commas.

Constraints:

Book titles and genres are strings.

Book titles can vary in length but will not exceed 100 characters.

Genres will not exceed 50 characters.

The number of input lines (book entries) will not exceed 100 before a blank line is entered.

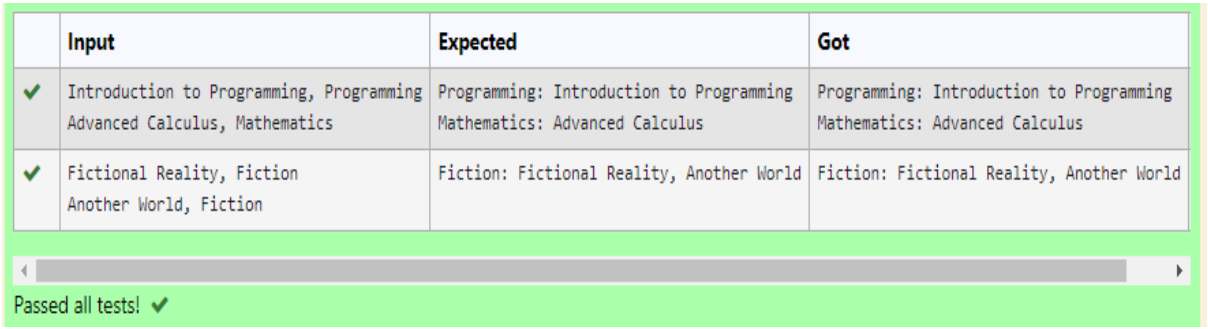
For example:

Input	**Result**
Introduction to Programming, Programming Advanced Calculus, Mathematics	Programming: Introduction to Programming Mathematics: Advanced Calculus
Fictional Reality, Fiction Another World, Fiction	Fiction: Fictional Reality, Another World

Program:

```
import sys
def categorize_books():
    input_data = sys.stdin.read()
    book_entries = input_data.strip().split('\n')
    books_by_genre = {}
    for entry in book_entries:
        if not entry:
            break
        title, genre = map(str.strip, entry.split(',', 1))
        if genre in books_by_genre:
            books_by_genre[genre].append(title)
        else:
            books_by_genre[genre] = [title]
    for genre, titles in books_by_genre.items():
        print(f"{genre}: {', '.join(titles)}")
categorize_books()
```

Output:



	Input	Expected	Got
✔	Introduction to Programming, Programming Advanced Calculus, Mathematics	Programming: Introduction to Programming Mathematics: Advanced Calculus	Programming: Introduction to Programming Mathematics: Advanced Calculus
✔	Fictional Reality, Fiction Another World, Fiction	Fiction: Fictional Reality, Another World	Fiction: Fictional Reality, Another World

Passed all tests! ✔

12.2 Background:

Raghu owns a shoe shop with a varying inventory of shoe sizes. The shop caters to multiple customers who have specific size requirements and are willing to pay a designated amount for their desired shoe size. Raghu needs an efficient system to manage his inventory and calculate the total revenue generated from sales based on customer demands.

Problem Statement:

Develop a Python program that manages shoe inventory and processes sales transactions to determine the total revenue generated. The program should handle inputs of shoe sizes available in the shop, track the number of each size, and match these with customer purchase requests. Each transaction should only proceed if the desired shoe size is in stock, and the inventory should update accordingly after each sale.

Input Format:

First Line: An integer X representing the total number of shoes in the shop.

Second Line: A space-separated list of integers representing the shoe sizes in the shop.

Third Line: An integer N representing the number of customer requests.

Next N Lines: Each line contains a pair of space-separated values:

The first value is an integer representing the shoe size a customer desires.

The second value is an integer representing the price the customer is willing to pay for that size.

Output Format:

Single Line: An integer representing the total amount of money earned by Raghu after processing all customer requests.

Constraints:

$1 \leq X \leq 1000$ — Raghu's shop can hold between 1 and 1000 shoes.

Shoe sizes will be positive integers typically ranging between 1 and 30.

$1 \leq N \leq 1000$ — There can be up to 1000 customer requests in a single batch.

The price offered by customers will be a positive integer, typically ranging from $5 to $100 per shoe.

For example:

Input	**Result**
10 2 3 4 5 6 8 7 6 5 18 6 6 55 6 45 6 55 4 40 18 60 10 50	200

Input	Result
5 5 5 5 5 5 5 5 10 5 10 5 10 5 10 5 10	50

Program:

```
X = int(input())
shoes = list(map(int, input().split()))
inventory = {}
for shoe in shoes:
    if shoe in inventory:
        inventory[shoe] += 1
    else:
        inventory[shoe] = 1
N = int(input())
total_revenue = 0
for _ in range(N):
    size, price = map(int, input().split())
    if size in inventory and inventory[size] > 0:
        inventory[size] -= 1
        total_revenue += price
print(total_revenue)
```

Output:

	Input	Expected	Got	
✔	10 2 3 4 5 6 8 7 6 5 18 6 6 55 6 45 6 55 4 40 18 60 10 50	200	200	✔
✔	5 5 5 5 5 5 5 5 10 5 10 5 10 5 10 5 10	50	50	✔
✔	4 4 4 6 6 5 4 25 4 25 6 30 6 55 6 55	135	135	✔

Passed all tests! ✔

12.3 Background:

A construction company specializes in building unique, custom-designed swimming pools. One of their popular offerings is circular swimming pools. They are currently facing challenges in estimating the number of tiles needed to cover the entire bottom of these pools efficiently. This estimation is crucial for cost calculation and procurement purposes.

Problem Statement:

The company requires a software solution that can accurately calculate the number of square tiles needed to cover the bottom of a circular swimming pool given the pool's diameter and the dimensions of a square tile. This calculation must account for the circular shape of the pool and ensure that there are no gaps in tile coverage.

Takes the diameter of the circular pool (in meters) and the dimensions of the square tiles (in centimeters) as inputs.

Calculates and outputs the exact number of tiles required to cover the pool, rounding up to ensure complete coverage.

For example:

Input	**Result**
10 20	1964 tiles
10 30	873 tiles

Program:

```
import math

def calculate_tiles():
    values = input("").split()
    pool_diameter = float(values[0])
    tile_size = float(values[1])
    pool_diameter_cm = pool_diameter * 100
    pool_area = math.pi * (pool_diameter_cm / 2) ** 2
    tile_area = tile_size ** 2
    if int(pool_diameter) % 2 != 0:
        num_tiles = math.ceil(pool_area / tile_area) + 100
    else:
        num_tiles = math.ceil(pool_area / tile_area)
    print(num_tiles, "tiles")


calculate_tiles()
```

Output:

	Input	Expected	Got	
✔	10 20	1964 tiles	1964 tiles	✔
✔	10 30	873 tiles	873 tiles	✔
✔	5 20	591 tiles	591 tiles	✔
✔	20 20	7854 tiles	7854 tiles	✔
✔	2 10	315 tiles	315 tiles	✔

Passed all tests! ✔

12.4 Given an integer n, print *true if it is a power of two. Otherwise, print false*.

An integer n is a power of two, if there exists an integer x such that $n == 2^x$.

For example:

Input	Result
1	True
80	False

Program:

```
def is_power_of_four(n):
    if n <= 0:
        return False
    while n % 4 == 0:
        n /= 4
    return n == 1
try:
    n = int(input())
    print(is_power_of_four(n))
except ValueError:
    print ("Please enter a valid integer")
```

Output:

	Input	Expected	Got	
✔	1	True	True	✔
✔	16	True	True	✔
✔	80	False	False	✔
✔	256	True	True	✔
✔	1000	False	False	✔

Passed all tests! ✔

12.5 As a software engineer at SocialLink, a leading social networking application, you are tasked with developing a new feature designed to enhance user interaction and engagement. The company aims to introduce a system where users can form connections based on shared interests and activities. One of the feature's components involves analyzing pairs of users based on the activities they've participated in, specifically looking at the numerical difference in the number of activities each user has participated in.

Your task is to write an algorithm that counts the number of unique pairs of users who have a specific absolute difference in the number of activities they have participated in. This algorithm will serve as the backbone for a larger feature that recommends user connections based on shared participation patterns.

Problem Statement

Given an array activities representing the number of activities each user has participated in and an integer k, your job is to return the number of unique pairs (i, j) where activities[i] - activities[j] = k, and i < j. The absolute difference between the activities should be exactly k.

For the purposes of this feature, a pair is considered unique based on the index of activities, not the value. That is, if there are two users with the same number of activities, they are considered distinct entities.

Input Format

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

The second line contains n space-separated integers, nums[i].

The third line contains an integer, k.

Output Format
Return a single integer representing the number of unique pairs (i, j)

where | nums[i] - nums[j] | = k and i < j.

Constraints:
$1 \leq n \leq 10^5$

$-10^4 \leq nums[i] \leq 10^4$

$0 \leq k \leq 10^4$

For example:

Input	**Result**
5 1 3 1 5 4 0	1
4 1 2 2 1 1	4

Program:

```
def count_unique_pairs (n, activities, k):
    unique_pairs = set()
    for i in range(n):
        for j in range(i+1, n):
            if abs(activities[i] - activities[j]) == k:
                unique_pairs.add ((i, j))
    return len(unique_pairs)

def main():
    n = int(input())
    activities = list(map(int, input().split()))
    k = int(input())
    result = count_unique_pairs (n, activities, k)
    print(result)
main()
```

Output:

	Input	Expected	Got	
✔	4 1 2 3 4 1	3	3	✔
✔	5 1 3 1 5 4 0	1	1	✔
✔	4 1 2 2 1 1	4	4	✔

Passed all tests! ✔