RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23221
PYTHON PROGRAMMING LAB

Laboratory Observation Note Book

Name : Omesh Balamurugan

Year / Branch / Section : I-st year / CSE / 'D'

Register No. : 230701222

Semester : II

Academic Year : 2023-2024

INDEX

Reg. No. : 230701222 Name : Omesh Balamurugan

Year : 2023-24 Branch : CSE Sec : D

S. No.	Date	Title	Page No.	Teacher's Signature / Remarks
Introduction to python-Variables-Datatypes-Input/Output-Formatting				
1.1		Converting Input Strings		
1.2		Gross salary		
1.3		Square Root		
1.4		Gain percent		
1.5		Deposits		
1.6		Carpenter		
Operators in Python				
2.1		Widgets and Gizmos		
2.2		Doll Sings		
2.3		Birthday party		
2.4		Hamming Weight		
2.5		Compound Interest		
2.6		Eligible to donate blood		
2.7		C or D		
2.8		Troy Battle		
2.9		Tax and Tip		
2.10		Return last digit of the given number		
Selection Structures in Python				
3.1		Admission eligibility		
3.2		Classifying triangles		

3.3		Electricity Bill		
3.4		IN/OUT		
3.5		Vowel or Constant		
3.6		Leap Year		
3.7		Month name to Days		
3.8		Pythagorean triple		
3.9		Second Last Digit		
3.10		Chinese Zodiac		
Algorithmic Approach: Iteration Control Structures				
4.1		Factors of a Number		
4.2		Non-Repeated Digits Count		
4.3		Prime Checking		
4.4		Next Perfect Square		
4.5		Nth Fibonacci		
4.6		Disarium Number		
4.7		Sum of Series		
4.8		Unique Digits Count		
4.9		Product of single digits		
4.10		Perfect Square After adding One		
Strings in Python				
5.1		Count chars		
5.2		Decompress the String		
5.3		First N Common Characters		
5.4		Remove Characters		
5.5		Remove Palindrome Words		
5.6		Return Second Word in Uppercase		
5.7		Reverse String		
5.8		String characters balance Test		
5.9		Unique Names		
5.10		Username Domain Extension		
List in Python				
6.1		Monotonic array		
6.2		Check pair with difference k .		

6.3		Count Elements		
6.4		Distinct Elements in an Array		
6.5		Element Insertion		
6.6		Find the Factor		
6.7		Merge list		
6.8		Merge Two Sorted Arrays Without Duplication		
6.9		Print Element Location		
6.10		Strictly increasing		
Tuples & Set				
7.1		Binary String		
7.2		Check Pair		
7.3		DNA Sequence		
7.4		Print repeated no		
7.5		Remove repeated		
7.6		malfunctioning keyboard		
7.7		American keyboard		
Dictionary				
8.1		Uncommon Words		
8.2		Sort Dictionary By Values Summation		
8.3		Winner Of Election		
8.4		Student Record		
8.5		Scramble Score		
Functions				
9.1		Abundant Number		
9.2		Automorphic number or not		
9.3		Check Product of Digits		
9.4		Christmas Discount		
9.5		Coin Change		
9.6		Difference Sum		

9.7		Ugly number		
Searching & Sorting				
10.1		Merge Sort		
10.2		Bubble Sort		
10.3		Peak Element		
10.4		Binary Search		
10.5		Frequency of Numbers		

01 - Introduction to Python-Variables-Datatypes

Input/Output-Formatting

Sample Output:

10,<class 'int'>

10.9,<class 'float'>

For example:

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

Ex. No. : 1.1 Date:

Register No.: Name:

Converting Input Strings

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

Sample Input:

10

10.9

```
A=int(input())
B=float(input())
Print(a,",",type(a),sep='')

Print(round(b,1),",",type(b),sep="")
```

Sample Input:

10000

Sample Output:

16000

For example:

Input	Result
10000	16000

Ex. No. : 1.2 Date:

Register No.: Name:

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.

```
basic_salary=int(input())

allow=basic_salary*(40/100)

hra=basic_salary*(20/100)

gross_salary=hra+allow+basic_salary

print(int(gross_salary))
```

Sample Input:

8.00

Sample Output:

2.828

For example:

Input	Result
14.00	3.742

Ex. No. : 1.3 Date:

Register No.: Name:

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.

```
Import math
A=float(input())
B=math.sqrt(a)
Print("{:.3f}".format(b))
```

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.

Ex. No. : 1.4 Date:

Register No.: Name:

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.

```
X=int(input())
Y=int(input())
Z=float(input())
T=x+y
Gain=z-t
Per=(gain/t)*100
Print(format(per,".2f"),"is the gain percent.")
```

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.

Ex. No. : 1.5 Date:

Register No.: Name:

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.

```
N=int(input())
M=int(input())
A=n*0.10
B=m*0.25
Total=a+b
Print(f"Your total refund will be ${total:.2f}.")
```

Sample Input:

450

Sample Output:

weekdays 10.38

weekend 0.38

For example:

Input	Result
450	weekdays 10.38 weekend 0.38

Ex. No. : 1.6 Date:

Register No.: Name:

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.

```
Z=int(input())
X=abs(z-500)/13
X=x/10
Wkd=x+10

Abc=abs(x)
Print(f"weekdays {wkd:.2f}")
Print(f"weekend {abc:.2f}")
```

02- Operators in Python

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.

Ex. No. : 2.1 Date:

Register No.: Name:

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.

A=int(input())

B=int(input())

Print("The total weight of all these widgets and gizmos is",(a*75+b*112),"grams.")

Sample Input

10

Sample Output

True

Explanation:

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

Ex. No. : 2.2 Date:

Register No.: Name:

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.

```
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)
```

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

Ex. No. : 2.3 Date:

Register No.: Name:

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.

```
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)
```

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.

Ex. No. : 2.4 Date:

Register No.: Name:

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)

```
A=bin(int(input()))

Print(a.count("1"))
```

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

Ex. No. : 2.5 Date:

Register No.: Name:

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.

.

```
a=float(input())
b1=a*1.04
b2=b1*1.04
b3=b2*1.04
print("Balance as of end of Year 1: ${:.2f}.".format(b1))
print("Balance as of end of Year 2: ${:.2f}.".format(b2))
print("Balance as of end of Year 3: ${:.2f}.".format(b3))
```

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

Ex. No. : 2.6 Date:

Register No.: Name:

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.

A=int(input())

B=int(input())

Print(a>=18 and b>40)

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

Ex. No. : 2.7 Date:

Register No.: Name:

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.

A=int(input())

Print(chr(a+67))

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

Ex. No. : 2.8 Date:

Register No.: Name:

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.

```
A=int(input())
B=int(input())
Print(a%3==0 and b%2==0)
```

Sample Input

100

Sample Output

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

Ex. No. : 2.9 Date:

Register No.: Name:

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.

```
a=int(input())
tax=a*(5/100)
tip=a*(18/100)
total=tax+tip+a
print("The tax is {:.2f} and the tip is {:.2f}, making the total {:.2f}".format(tax,tip,total))
```

For example:

Input	Result
123	3

Ex. No. : 2.10 Date:

Register No.: Name:

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

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

03 - Selection Structures in Python

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
50 80 80	The candidate is eligible

Ex. No. : 3.1 Date:

Register No.: Name:

Admission 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

```
A=int(input())
B=int(input())
C=int(input())
Total=a+b+c
If(a>=65 and b>=55 and c>=50):
   Print('The candidate is eligible')
Elif(total>=180):
   Print("The candidate is eligible")
Else:
   Print("The candidate is not eligible")
```

Sample Input 1

60

60

60

Sample Output 1

That's a equilateral triangle

For example:

Input	Result
40 40 80	That's a isosceles triangle

Ex. No. : 3.2 Date:

Register No.: Name:

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.

```
A=int(input())

B=int(input())

C=int(input())

If(a==b==c):

    Print("That's a equilateral triangle")

Elif(a==b):

    Print("That's a isosceles triangle")

Else:

    Print("That's a scalene triangle")
```

Sample Test Cases

Test Case 1

Input

50

Output

100.00

Test Case 2

Input

300

Output

517.50

For example:

Input	Result
500	1035.00

Ex. No. : 3.3 Date:

Register No.: Name:

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/-

```
a=float(input())
if a<=199:
   bill=a*1.20
if 200<=a<400:
   bill=a*1.50
if 400<=a<600:
   bill=(a*1.80)
if a>600:
   bill=(a*2.00)
if bill>400:
   bill+=bill*0.15
if bill<100:
   bill=100.00
```

```
print("{:.2f}".format(bill))
```

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

Ex. No. : 3.4 Date:

Register No.: Name:

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.

```
A=int(input())

B=int(input())

C=a/2

If(c<b):

   Print('IN')

Else:

   Print("OUT")
```

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 Input3

c

Sample Output 3

It's a consonant.

For example:

Input	Result
y	Sometimes it's a vowel... Sometimes it's a consonant.
u	It's a vowel.
p	It's a consonant.

Ex. No. : 3.5 Date:

Register No.: Name:

Vowel or Consonant

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.

```
ai=input()

If ai==’a’:

   Print(“It’s a vowel.”)

Elif ai==’e’:

   Print(“It’s a vowel.”)

Elif ai==’I’:

   Print(“It’s a vowel.”)

Elif ai==’o’:

   Print(“It’s a vowel.”)

Elif ai==’u’:

   Print(“It’s a vowel.”)

Elif ai==’y’:

   Print(“Sometimes it’s a vowel… Sometimes it’s a consonant.”)

Else:

   Print(“It’s a consonant”)
```

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.

Ex. No. : 3.6 Date:

Register No.: Name:

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.

```
A=int(input())

If (a%400==0):

   Print(a,"is a leap year.")

Elif (a%100==0):

   Print(a,"is not a leap year.")

Else:

   Print(a,"is a leap year.")
```

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.
March	March has 31 days in it.

Ex. No. : 3.7 Date:

Register No.: Name:

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.

```
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.")
```

Sample Input

3

5

4

Sample Output

Yes

For example:

Input	Result
3 4 5	Yes

Ex. No. : 3.8 Date:

Register No.: Name:

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".

```
A=int(input())
B=int(input())
C=int(input())
If(a*a+b*b==c*c):
   Print(‘yes’)
Elif(b*b+c*c==a*a):
   Print(‘yes’)
Elif(c*c+a*a==b*b):
   Print(‘yes’)
Else:
   Print(‘no’)
```

For example:

Input	Result
197	9

Ex. No. : 3.9 Date:

Register No.: Name:

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.

```
a=(int(input()))
a=abs(a)

if(0<=a<=9):
   print("-1")
else:
   rem=(a/10)%10
   print(int(rem))
```

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.

Ex. No. : 3.10 Date:

Register No.: Name:

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.

```
A=int(input())

R=a%12

If r==0

   Print(a,"is the year of the Monkey.")

Elif r==1:

   Print(a,"is the year of the Rooster.")
```

```
Elif r==2:
    Print(a,”is the year of the Dog.”)
Elif r==3:
    Print(a,”is the year of the Pig.”)
Elif r==4:
    Print(a,”is the year of the Rat.”)
Elif r==5:
    Print(a,”is the year of the Ox.”)
Elif r==6:
    Print(a,”is the year of the Tiger.”)
Elif r==7:
    Print(a,”is the year of the Hare.”)
Elif r==8:
    Print(a,”is the year of the Dragon.”)
Elif r==9:
    Print(a,”is the year of the Snake.”)
Elif r==10:
    Print(a,”is the year of the Horse.”)
Else:
    Print(a,”is the year of the Sheep.”)
```

04 - Iteration Control Structures

For example:

Input	Result
20	1 2 4 5 10 20

Ex. No. : 4.1 Date:

Register No.: Name:

Factors of a number

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

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

For example:

Input	Result
292	1
1015	2
108	3
22	0

Ex. No. : 4.2 Date:

Register No.: Name:

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.

```
number = int(input("Enter a number to find the count of non-repeated digits: "))
num_str = str(number)
non_repeated_digits = []

for digit in num_str:
    if num_str.count(digit) == 1:
        non_repeated_digits.append(int(digit))

non_repeated_count = len(non_repeated_digits)
```

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

Ex. No. : 4.3 Date:

Register No.: Name:

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.

```
n=int(input())
flag=0
for i in range(2,n):
    if(n%2)==0:
        flag=1
        break
    if(n%i)==0:
        flag=1
        break

if flag:
    print("1")
else:
    print("2")
```

Input Format:

Integer input from stdin.

Output Format:

Perfect square greater than N.

Example Input:

10

Output:

16

Ex. No. : 4.4 Date:

Register No.: Name:

Next Perfect Square

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

```
num=int(input())
while 1:
    num=num+1
    root=(num**0.5)
    if root==int(root):
        print(num)
        break
```

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

Ex. No. : 4.5 Date:

Register No.: Name:

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.

```
n=int(input())
a,b=0,1


for i in range(1,n):
 a,b=b,a+b

print(a)
```

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

Ex. No. : 4.6 Date:

Register No.: Name:

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.

```
number = int(input())
num_str = str(number)
length = len(num_str)
total = 0

for i in range(length):
  total += int(num_str[i]) ** (i + 1)


if total==number:
   print("yes")
else:
   print(" no")
```

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

Ex. No. : 4.7 Date:

Register No.: Name:

Sum of Series

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)

```
n=int(input())

total=0

term=0


for i in range(1,n+1):

  term=(term*10)+1

  total=total+term
```

For example:

Input	Result
292	2
1015	3

Ex. No. : 4.8 Date:

Register No.: Name:

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'.

```
number = int(input())
num_str = str(number)
unique_digits = set()

for digit in num_str:
    unique_digits.add(digit)

unique_digit_count = len(unique_digits)

print( unique_digit_count)
```

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

Ex. No. : 4.9 Date:

Register No.: Name:

Product of single digit

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

```
number = int(input())

if number < 10:
    print("yes")
else:
    for factor in range(2, 10):
        if number % factor == 0 and 1 <= number // factor <= 9:
            print("yes")
            break
    else:
        print("no")
```

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

Ex. No. : 4.10 Date:

Register No.: Name:

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.

```
num=int(input())
num=num+1
a=int(num**0.5)
if a**2==num:
    print("yes")
else:
    print("no")
```

05 - List in Python

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 3 3	2
3 1 2 1	1

Ex. No. : 5.1 Date:

Register No.: Name:

Balanced Array

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 $0 \leq i < n$.

```
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))
```

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

Ex. No. : 5.2 Date:

Register No.: Name:

Check pair with difference k

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.

```
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)
```

Sample Test Cases

Test Case 1

Input

7

23

45

23

56

45

23

40

Output

23 occurs 3 times

45 occurs 2 times

56 occurs 1 times

40 occurs 1 times

Ex. No. : 5.3 Date:

Register No.: Name:

Count Elements

Complete the program to count frequency of each element of an array. Frequency of a particular element will be printed once.

```
a=int(input())
dic={}
for i in range(a):
    k=int(input())
    if k not in dic:
      dic[k]=1
    else:
      dic[k]+=1
for i in dic:
    print(f"{i} occurs {dic[i]} times")
```

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 2 3 3	1 2 3

Ex. No. : 5.4 Date:

Register No.: Name:

Distinct Elements in an Array

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 take 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

```
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=' ')
```

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

Ex. No. : 5.5 Date:

Register No.: Name:

Element Insertion

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.

```
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)
```

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

Ex. No. : 5.6 Date:

Register No.: Name:

Find the Factor

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the pth element of the list, sorted ascending. If there is no pth element, return 0.

Constraints

$1 \leq n \leq 1015$

$1 \leq p \leq 109$

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.

```
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)
```

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]]

Ex. No. : 5.7 Date:

Register No.: Name:

Merge List

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

```
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)
```

Sample Input 1

5
1
2
3
6
9
4
2
4
5
10

Sample Output 1

1 2 3 4 5 6 9 10

Ex. No. : 5.8 Date:

Register No.: Name:

Merge Two Sorted Arrays Without Duplication

Output is a merged array without duplicates.

```
N1 = int(input())
array1 = [int(input()) for _ in range(N1)]
N2 = int(input())
array2 = [int(input()) for _ in range(N2)]
merged_array = array1 + array2

merged_array = list(set(merged_array))
merged_array.sort()
print(*merged_array)
```

Input Format
N1 - no of elements in array 1
Array elements for array 1
N2 - no of elements in array 2
Array elements for array2
Output Format
Display the merged array

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.

Ex. No. : 5.9 Date:

Register No.: Name:

Print Element Location

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.

```
 num_elements = int(input())
lst = []
for _ in range(num_elements):
    lst.append(input())
element = input()
count = 0
for i in range(num_elements):
    if lst[i] == element:
        print(f"{element} is present at location {i + 1}.")
        count += 1
print(f"{element} is present {count} times in the array.")
```

Sample Test Case

Input

7

1

2

3

0

4

5

6

Output

True

Ex. No. : 5.10 Date:

Register No.: Name:

Strictly increasing

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"

```
def check_increasing_or_decreasing(lst):
    # Function to check if a list is strictly increasing or strictly decreasing
    increasing = True
    decreasing = True
    for i in range(1, len(lst)):
        if lst[i] > lst[i - 1]:
            decreasing = False
        elif lst[i] < lst[i - 1]:
            increasing = False
    return increasing or decreasing


def check_strictly_increasing_with_removal(lst):
    # Function to check if removing only one element makes the list strictly increasing or decreasing
    for i in range(len(lst)):
        temp_lst = lst[:i] + lst[i+1:]
        if check_increasing_or_decreasing(temp_lst):
            return True
```

```
    return False

# Input
n = int(input())
lst = []
for _ in range(n):
    lst.append(int(input()))

# Check if the list is strictly increasing or decreasing
if check_increasing_or_decreasing(lst) or check_strictly_increasing_with_removal(lst):
    print("True")
else:
    print("False")
```

06 - Strings in Python

For example:

Input Result
rec@123
3
3
1

Ex. No. : 6.1 Date:

Register No.: Name:

Count Chars

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

```
s=input()
l=n=c=0
for i in s:
    if(i.isalpha()):
        l+=1
    elif(i.isdecimal()):
        n+=1
    else:
        c+=1
print(l)
print(n)
print(c)
```

Sample Input 1
a2b4c6

Sample Output 1
aabbbbccccccc

Ex. No. : 6.2 Date:

Register No.: Name:

Decompress the String

Assume that the given string has enough memory. Don't use any extra space(IN-PLACE)

```
test_string = input()

res = []
numbers = []

temp = ""

for i in test_string:
    if i.isnumeric():
        temp += i
    else:
        if temp!= "":
            numbers.append(int(temp))
            temp = ""
        res.append(i)

if temp!= "":
    numbers.append(int(temp))

for i in range(0,len(res)):
    print(str(res[i]*numbers[i]),end="")
```

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.

Ex. No. : 6.3 Date:

Register No.: Name:

First N Common Chars

Two 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.

```
def common_characters(S1, S2, N):
    common_chars = []
    for char in S1:
        if char in S2 and char not in common_chars:
            common_chars.append(char)
            if len(common_chars) == N:
                break
    return ''.join(common_chars)
S1 = input().strip()
S2 = input().strip()
N = int(input().strip())
print(common_characters(S1, S2, N))
```

Sample Input 1
experience
enc

Sample Output 1
xpri

Ex. No. : 6.4 Date:

Register No.: Name:

Remove Characters

Given two Strings s1 and s2, remove all the characters from s1 which is present in s2.

Constraints
1<= string length <= 200

```
s1=input()
s2=input()
s1=''.join(i for i in s1 if i not in s2)
print(s1)
```

For example:

Input	Expected
Malayalam is my mother tongue	is my mother tongue
He did a good deed	he good

Ex. No. : 6.5 Date:

Register No.: Name:

Remove Palindrome Words

String should contain only the words are not palindrome.

Sample Input 1
Malayalam is my mother tongue

Sample Output 1
is my mother tongue

```
l=input().lower()
s1=l.split()
for i in s1:
    if(i[::-1]!=i):
        print(i,end=' ')
```

For example:

Input Result
Wipro Technologies Bangalore
TECHNOLOGIES
Hello World
WORLD
Hello
LESS

Ex. No. : 6.6 Date:

Register No.: Name:

Return Second World in Uppercase

Write a program that takes as input a string (sentence), and returns its second word in uppercase.

For example:

If input is “Wipro Technologies Bangalore” the function should return “TECHNOLOGIES”
If input is “Hello World” the function should return “WORLD”
If input is “Hello” the program should return “LESS”

NOTE 1: If input is a sentence with less than 2 words, the program should return the word “LESS”.
NOTE 2: The result should have no leading or trailing spaces.

```
s=input().split()
if(len(s)>=2):
    print(s[1].upper())
else:
    print("LESS")
```

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

For example:

Input	Result
A&x#	
x&A#	

Ex. No. : 6.7 Date:

Register No.: Name:

Revers String

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.

```
seq =input()


chars = ['$', '%', '*', '#', '^']

nums = []


for i in range(len(seq)):

    if seq[i] not in chars:

        nums.append(seq[i])


nums.reverse()


for j in seq:

    if j in chars:

        idx = seq.index(j)

        nums.insert(idx, j)


reverse = "".join(nums)


print(reverse)
```

For example:

Input Result
Yn
PYnative
True

Ex. No. : 6.8 Date:

Register No.: Name:

String characters balance Test

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".

```
def balance(s1,s2):
    flag=True
    for i in s1:
        if i in s2:
            pass
        else:
            flag=False
    return flag
s1=input()
s2=input()
if balance(s1,s2)==True:
    print("True")
else:
    print("False")
```

Input:
first
second
first
third
second

then your program should display:

Output:
first
second
third

Ex. No. : 6.9 Date:

Register No.: Name:

Unique Names

In this exercise, you will create a program that reads words from the user until the user enters a blank line. After the user enters a blank line your program should display each word entered by the user exactly once. The words should be displayed in the same order that they were first entered. For example, if the user enters:

```
l=[]
try:
  while True:
    s=input()
    if s not in l:
      l.append(s)
except EOFError:
  print('\n'.join(l))
```

Example Input/Output 1:

Input:

vijayakumar.r@rajalakshmi.edu.in

Output:

edu.in
rajalakshmi
vijayakumar.r

Ex. No. : 6.10 Date:

Register No.: Name:

Username Domain Extension

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

```
s=input()
s1=s.find(".")
s2=s.find("@")
print(s[s1+1:])
print(s[s2+1:s1])
print(s[:s2])
```

07 - Functions

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

Ex. No. : 7.1 Date:

Register No.: Name:

Abundant Number

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

```
def abundant(n):
    sum1=0
    for i in range(1,(n//2)+1):
        if(n%i==0):
            sum1+=i
    if(sum1>n):
        return "Yes"
    else:
        return "No"
```

Input Format:
Take a Integer from Stdin
Output Format:
Print Automorphic if given number is Automorphic number, otherwise Not Automorphic
Example input: 5 Output: Automorphic Example input: 25 Output: Automorphic Example input: 7 Output: Not Automorphic
For example:

Test	Result
print(automorphic(5))	Automorphic

Ex. No. : 7.2 Date:

Register No.: Name:

Automorphic number or not

An automorphic number is a number whose square ends with the number itself. For example, 5 is an automorphic number because 5*5 =25. The last digit is 5 which same as the given number.

If the number is not valid, it should display “Invalid input”.
If it is an automorphic number display “Automorphic” else display “Not Automorphic”.

```
def automorphic(n):
    if n <= 0:
        return "Invalid input"  # Only positive integers are valid

    square = n * n
    if str(square).endswith(str(n)):
        return "Automorphic"
    else:
        return "Not Automorphic"
```

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

Ex. No. : 7.3 Date:

Register No.: Name:

Check Product of Digits

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.

```
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
```

Input

The input consists of an integer orderValue, representing the total bill amount.

Output

Print an integer representing the discount value for the given total bill amount.

Example Input

578

Output

12

For example:

Test	Result
print(christmasDiscount(578))	12

Ex. No. : 7.4 Date:

Register No.: Name:

Christmas Discount

An e-commerce company plans to give their customers a special discount for Christmas. They are planning to offer a flat discount. The discount value is calculated as the sum of all the prime digits in the total bill amount.

Write an python code to find the discount value for the given total bill amount.

Constraints

$1 <= orderValue < 10e^{100000}$

```
def christmasDiscount(total_bill):
    prime_digits = {'2', '3', '5', '7'}
    total_bill_str = str(total_bill)

    discount_value = 0

    for digit in total_bill_str:
        if digit in prime_digits:
            discount_value += int(digit)

    return discount_value
total_bill = 75320
discount = christmasDiscount(total_bill)
```

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

Ex. No. : 7.5 Date:

Register No.: Name:

Coin Change

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

```
 return countdef coinChange(n):
count=0
while n>0:
    if(n>=4):
        n-=4
    elif(n==3):
        n-=3
    elif(n==2):
        n-=2
    elif(n==1):
        n-=1
    count+=1
```

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

Ex. No. : 7.6 Date:

Register No.: Name:

Difference Sum

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

```
def differenceSum(n):
    s1=str(n)
    s_e=0
    s_o=0
    for i, digit in enumerate(s1):
        if(i+1)%2==0:
            s_e+=int(digit)
        else:
            s_o+=int(digit)
    return abs(s_e - s_o)

n=1234567890
d=differenceSum(n)
```

For example:

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

Ex. No. : 7.7 Date:

Register No.: Name:

Ugly number

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.

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

    # Divide the number by 5 until it's no longer divisible by 5
    while n % 5 == 0:
        n //= 5

    # If after dividing by 2, 3, and 5, the number becomes 1, it's ugly
    if n == 1:
        return "ugly"
    else:
        return "not ugly"
```

08 – Tuple/Set

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

For example:

Input	Result
01010101010	Yes
010101 10101	No

Ex. No. : 8.1 Date:

Register No.: Name:

Binary String

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.

```
a=input()

c=0

for i in range(len(a)):

   if (a[i]=='0' or a[i]=='1'):

      c=c+1


if c==len(a):

   print("Yes")

else:

   print("No")
```

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

Ex. No. : 8.2 Date:

Register No.: Name:

Check Pair

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**.

```
tuple_str = input()
t = tuple(map(int, tuple_str.split(",")))
K = int(input())

distinct_pairs = set()
for i in range(len(t)):
    for j in range(i + 1, len(t)):
        if t[i] + t[j] == K:
            distinct_pairs.add((min(t[i], t[j]), max(t[i], t[j])))

print(len(distinct_pairs))
```

Example 1:

Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
Output: ["AAAAACCCCC","CCCCCAAAAA"]

Example 2:

Input: s = "AAAAAAAAAAAAA"
Output: ["AAAAAAAAAA"]

For example:

Input	Result
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA

Ex. No. : 8.3 Date:

Register No.: Name:

DNA Sequence

The **DNA sequence** is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'.

For example, "ACGAATTCCG" is a **DNA sequence**.

When studying **DNA**, it is useful to identify repeated sequences within the DNA.

Given a string s that represents a **DNA sequence**, return all the **10-letter-long** sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in **any order**.

```
dna = input()
if not all(char in "ACGT" for char in dna):
  exit()
rep = []
d = {}
for i in range(len(dna) - 9):
  sub = dna[i:i+10]
  if sub in d:
    if d[sub] == 1:
      rep.append(sub)
    d[sub] += 1
  else:
    d[sub] = 1

if rep:
  for sequence in rep:
   print(sequence)
```

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

Ex. No. : 8.4 Date:

Register No.: Name:

Print repeated no

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.

```
a = input()
b = a.split()  # Call split as a function

seen = set()

for num in b:
    if num in seen:
        print(num)
    else:
        seen.add(num)
```

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

Ex. No. : 8.5 Date:

Register No.: Name:

Remove repeated

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.

```
a = input().split()
n1 = input().split()
n2 = input().split()
c = set()
for num in n1:
    if num not in n2:
        c.add(num)
for num1 in n2:
    if num1 not in n1:
        c.add(num1)
c = sorted(map(int, c))
if c:
 print(" ".join(map(str, c)))
 print(len(c))
```

```
else:
 print("NO SUCH ELEMENTS")
```

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

Ex. No. : 8.6 Date:

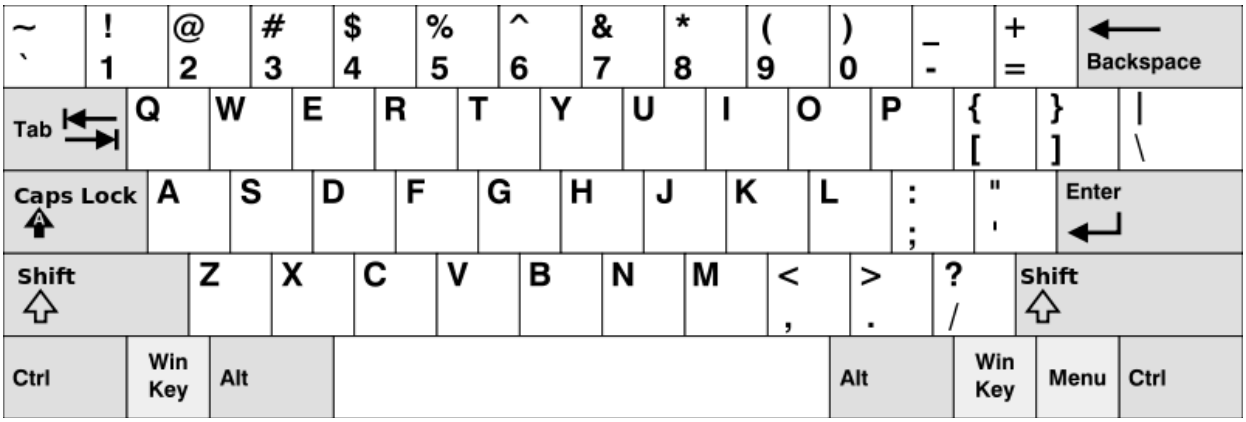
Register No.: Name:

Malfunctioning Keyboard

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.

```
text = input()

brokenLetters = set(input())


words = text.split()

count = len(words)


for word in words:

    if any(letter in brokenLetters for letter in word):

        count -= 1


print(count)
```



Example 1:

Input: words = ["Hello","Alaska","Dad","Peace"]
Output: ["Alaska","Dad"]

Example 2:

Input: words = ["omk"]
Output: []

Example 3:

Input: words = ["adsdf","sfd"]
Output: ["adsdf","sfd"]

For example:

Input	Result
4 Hello Alaska Dad Peace	Alaska Dad

Ex. No. : 8.7 Date:

Register No.: Name:

American keyboard

Given an array of strings words, return *the words that can be typed using letters of the alphabet on only one row of American keyboard like the image below.*

In the **American keyboard**:

- the first row consists of the characters "qwertyuiop",
- the second row consists of the characters "asdfghjkl", and
- the third row consists of the characters "zxcvbnm".

```
words_input = input()

words = words_input.split()



first_row = set("qwertyuiop")

second_row = set("asdfghjkl")

third_row = set("zxcvbnm")

result = []

for word in words:

    lowercase_word = word.lower()

    if lowercase_word[0] in first_row:

        row_set = first_row

    elif lowercase_word[0] in second_row:

        row_set = second_row

    else:
```

```
        row_set = third_row
    if all(char in row_set for char in lowercase_word):
        result.append(word)
print(result)
```

09 – Dictionary

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

Ex. No. : 9.1 Date:

Register No.: Name:

Uncommon words

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.

```
# Get input from the user for the two sentences
s1 = input()
s2 = input()

# Combine the words from both sentences into a single list
all_words = s1.split() + s2.split()

# Initialize an empty dictionary to store the word count
word_count = {}

# Count the occurrences of each word
for word in all_words:
    word_count[word] = word_count.get(word, 0) + 1

# Print the uncommon words separately

for word, count in word_count.items():
    if count == 1:
        print(word,end=" ")
```

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

Ex. No. : 9.2 Date:

Register No.: Name:

Sort Dictionary by Values Summation

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

```
n = int(input())
test_dict = {}
for _ in range(n):
    key, *values =input().split()
    test_dict[key] = list(map(int, values))
sorted_dict = {k: sum(v) for k, v in sorted(test_dict.items(), key=lambda item:sum(item[1]))}
for key, value in sorted_dict.items():
    print(f"{key} {value}")
```

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

For example:

Input	Result
10 John John Johny Jamie Jamie Johny Jack Johny Johny Jackie	Johny

Ex. No. : 9.3 Date:

Register No.: Name:

Winner of Election

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.

```
vote_count = {}

num_candidates = int(input())

for i in range(num_candidates):

    candidate_name = input()

    vote_count[candidate_name] = vote_count.get(candidate_name, 0) + 1

max_votes = max(vote_count.values())

max_vote_candidates = [candidate for candidate, votes in vote_count.items() if votes ==
max_votes]



# Determine the winner by selecting the lexicographically smallest candidate name

winner = min(max_vote_candidates)



# Print the winner

print(winner)
```

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

Ex. No. : 9.4 Date:

Register No.: Name:

Student Record

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

```
d={}
n=int(input())
for i in range (n):
    b=input().split(" ")
    l=[]
    for j in range (1,4):
        l.append(int(b[j]))
    d[b[0]]=l
max1=0
min1=101
maxh=0
minb=101
for i in d:
    b=sum(d[i])/4
    if(b>max1):
        max1=b
```

```
        name1=i
    if(b<min1):
        min1=b
        name2=i
    if(maxh<d[i][1]):
        maxh=d[i][1]
        maxhn=i
    if(minb>d[i][2]):
        minb=d[i][2]
        minbn=i
d1={}
l1,l2,l3,l4=[],[],[],[]
for i in d:
    b=sum(d[i])/4
    if(max1==b):
        l1.append(i)
    if(maxh==d[i][1]):
        l2.append(i)
    if(minb==d[i][2]):
        l3.append(i)
    if(min1==b):
        l4.append(i)
d1[1]=sorted(l1)
d1[2]=sorted(l2)
d1[3]=sorted(l3)
d1[4]=sorted(l4)
for i in d1:
    l=len(d1[i])
```

```
for j in range(l):
    print(d1[i][j],end=" ")
print()
```

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

Sample Input

REC

Sample Output

REC is worth 5 points.

Ex. No. : 9.5 Date:

Register No.: Name:

Scramble Score

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.
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.

```
letter_values = {
    'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R': 1, 'S': 1, 'T': 1, 'U': 1,
    'D': 2, 'G': 2,
    'B': 3, 'C': 3, 'M': 3, 'P': 3,
    'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
    'K': 5,
    'J': 8, 'X': 8,
    'Q': 10, 'Z': 10
}

word = input()
word = word.upper()
score = sum(letter_values.get(letter, 0) for letter in word)
print(f"{word} is worth {score} points.")
```

10 - Searching & Sorting

For example:

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

Ex. No. : 10.1 Date:

Register No.: Name:

Merge Sort

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

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    # Split the array into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    # Recursive calls to sort each half
    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    # Merge the sorted halves
    sorted_arr = []
    i = j = 0
    while i < len(left_half) and j < len(right_half):
```

```
        if left_half[i] < right_half[j]:
            sorted_arr.append(left_half[i])
            i += 1
        else:
            sorted_arr.append(right_half[j])
            j += 1

    # Add remaining elements from both halves
    sorted_arr.extend(left_half[i:])
    sorted_arr.extend(right_half[j:])
    return sorted_arr

# Input
n = int(input())
arr = list(map(int, input().split()))
# Sorting
sorted_arr = merge_sort(arr)

# Output
print(*sorted_arr)
```

Input Format

The first line contains an integer,n , the size of the list a . The second line contains n, space-separated integers a[i].

Constraints

· 2<=n<=600

· $1<=a[i]<=2x10^6$.

Output Format

You must print the following three lines of output:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

Sample Input 0

3

1 2 3

Sample Output 0

List is sorted in 0 swaps.

First Element: 1

Last Element: 3

For example:

Input	Result
3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3
5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9

Ex. No. : 10.2 Date:

Register No.: Name:

Bubble Sort

Given an listof integers, sort the array in ascending order using the *Bubble Sort* algorithm above. Once sorted, print the following three lines:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

For example, given a worst-case but small array to sort: a=[6,4,1]. It took 3 swaps to sort the array. Output would be

Array is sorted in 3 swaps.
First Element: 1
Last Element: 6

```
def bubble_sort(arr):

    n = len(arr)

    num_swaps = 0

    for i in range(n):

        for j in range(0, n-i-1):

            if arr[j] > arr[j+1]:

                arr[j], arr[j+1] = arr[j+1], arr[j]

                num_swaps += 1

    return arr, num_swaps


num_elements = int(input().strip())

array = list(map(int, input().strip().split()))
```

```
sorted_array, num_swaps = bubble_sort(array)


print(f"List is sorted in {num_swaps} swaps.")
print(f"First Element: {sorted_array[0]}")
print(f"Last Element: {sorted_array[-1]}")
```

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

Ex. No. : 10.3 Date:

Register No.: Name:

Peak Element

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]

```
def find_peak_element(nums):
    def find_peak_util(nums, low, high):
        mid = low + (high - low) // 2

        # Check if mid is a peak element
        if (mid == 0 or nums[mid] >= nums[mid - 1]) and (mid == len(nums) - 1 or nums[mid] >= nums[mid + 1]):
            return mid

        # If the left neighbor is greater, there must be a peak element on the left side
        if mid > 0 and nums[mid - 1] > nums[mid]:
            return find_peak_util(nums, low, mid - 1)

        # If the right neighbor is greater, there must be a peak element on the right side
        return find_peak_util(nums, mid + 1, high)
```

```
    return find_peak_util(nums, 0, len(nums) - 1)


# Get input from user
user_input = input("Enter a list of numbers separated by spaces: ")
nums = list(map(int, user_input.split()))


# Find peak element
peak_index = find_peak_element(nums)
print(f"T{peak_index} and the value is {nums[peak_index]}")
```

For example:

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

Ex. No. : 10.4 Date:

Register No.: Name:

Binary Search

Write a Python program for binary search.

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1

    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return True
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return False

sorted_list = list(map(int, input().split(',')))
target = int(input())

sorted_list.sort()
```

```
result = binary_search(sorted_list, target)

print(result)
```

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

Ex. No. : 10.5 Date:

Register No.: Name:

Frequency of Elements

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

Constraints:

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

```
def count_frequencies(arr):
    frequency_dict = {}
    for num in arr:
        if num in frequency_dict:
            frequency_dict[num] += 1
        else:
            frequency_dict[num] = 1
    sorted_keys = sorted(frequency_dict.keys())
    for key in sorted_keys:
        print(key, frequency_dict[key])

# Read input from the user
input_list = list(map(int, input().split()))

# Count frequencies and display the result
count_frequencies(input_list)
```