Fall 20-21 **CSE1001 Problem Solving and Programming**

Instructor: Oswald C **Student: Aditya Kumar Gupta**

This document contains a collection of problems taken from various sources and our solutions in Python programming. This is not a unique solution and we encourage you to try for other solutions as well. Thank You.

NUMBER THEORY AND ARITHMETIC PROGRAMS

(1) Products of Number except i

--Book by Charles Dierbach #1

Given an array of integers, return a new array such that each element at index `i` of the new array is the product of all the numbers in the original array except the one at `i`.

For example, if our input was `[1, 2, 3, 4, 5]`, the expected output would be `[120, 60, 40, 30, 24]`. If our input was `[3, 2, 1]`, the expected output would be `[2, 3, 6]`.

Follow-up: what if you can't use division?

Code-

```
def p(a):
    l=[]
    for I in range(len(a)):
        pro=1
        for j in range(len(a)):
            if j!=i:
                pro*=a[j]
        l.append(pro)
    return l


a=list(map(int, input().split()))
print(p(a))
```

(2) Passing Element

-Book by Charles Dierbach #2

Given a list of numbers and a number `k`, return whether any two numbers from the list add up to `k`.

For example, given `[10, 15, 3, 7]` and `k` of `17`, return true since `10 + 7` is `17`.

Bonus: Can you do this in one pass?

Code-

```
def e(l,k):
    for i in l:
        if k-i in l:
            print(bool(1))
            break
l,k=list(map(int, input().split())),int(input())
e(l,k)
```

(3) Flip A Coin

-Book by Maureen Sprankle & Jim Hubbard #1

Develop a solution to flip a coin a given amount of times and then print the number of heads and the number of tails. The equation to toss the coin is

$$Coin = Integer(Random * 2) + 1$$

When Coin = 1 the toss is heads, and when Coin = 2 the toss is tails. Random returns a number between 0 and 1, including 0 but not 1. Therefore, when Random is less than 0.5, Coin will equal 1: and when Random is greater than or equal to 0.5 and less than 1, Coin will equal 2.

Code-

```
from random import random

n=int(input("enter the no. of times you want to flip the coin : "))
h,t=0,0

for i in range(n):
    f=int((random()*2)+1)
    if f==1:
        h+=1
    else:
        t+=1

print("the total no. of heads in ",n," flips are ",h)
print("the total no. of tails in ",n," flips are ",t)
```

(4) Python Evaluation

-HackerRank #1

The `eval()` expression is a very powerful built-in function of Python. It helps in evaluating an expression. The expression can be a Python statement, or a code object.

For example:

```
>>> eval("9 + 5")
14
>>> x = 2
>>> eval("x + 3")
5
```

Here, `eval()` can also be used to work with Python keywords or defined functions and variables. These would normally be stored as strings.

For example:

```
>>> type(eval("len"))
<type 'builtin_function_or_method'>
```

Without eval()

```
>>> type("len")
<type 'str'>
```

Task

You are given an expression in a line. Read that line as a string variable, such as *var*, and print the result using *eval(var)*.

NOTE: Python2 users, please import `from __future__ import print_function`.

Constraint

Input string is less than 100 characters.

Sample Input

print(2 + 3)

Sample Output

5

Code-

```
a=input()
eval(a)
```

(5) Palindrome Number

--Book by Maureen Sprankle & Jim Hubbard #2

Write a solution to tell the user whether a number is a palindrome. (A palindrome is a number that is the same written both forward and backward.)

Code-

```
num=int(input("Enter a number : "))
n=num
res=0
while num>0:
 rem=num%10
 res=rem+res*10
 num=num//10
if res==n:
  print("Number is Palindrome")
else:
  print("Number is not Palindrome")
```

(6) Perfect Number

--Book by Charles Dierbach #3

Write a code to check that the given no. is Perfect no. or not. Perfect number, a positive integer that is equal to the sum of its proper divisors. The smallest perfect number is 6, which is the sum of 1, 2, and 3. Other perfect numbers are 28, 496, and 8,128.

If it's a perfect no. your code should return True, otherwise False.

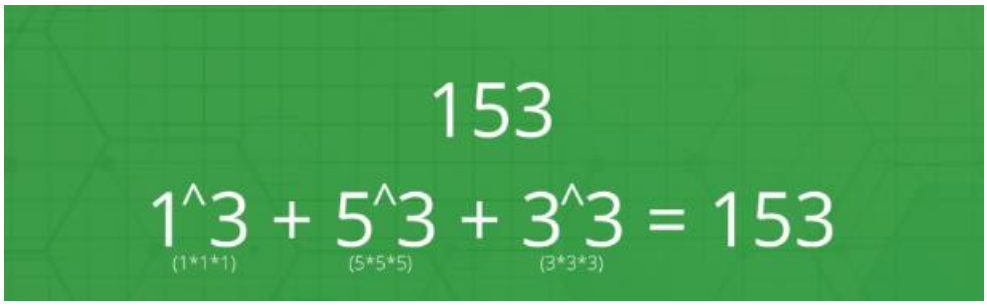
Code-

```
def p(n):
    a=n//2
    l=[]
    for i in range(1,a+1):
        if n%i==0:
            l.append(i)
        else:

            continue

    if n==sum(l):
        return bool(1)
    else:
        return bool(0)


print(p(int(input())))
```

(7) Armstrong Number

--Book by Charles Dierbach #4

Write a code to check Armstrong number. Armstrong number is a number that is equal to the sum of powers of total no. of digits of its digits. For example 0, 1, 153, 370, 371 and 407 are the Armstrong numbers.



Return True or False.

Code-

```
def x(n):
    y=list(str(n))
    a=len(y)
    c=0
    for i in y:
        c+=int(i)**a
    if c==n:
        return bool(1)
    else:
        return bool(0)


print(x(int(input())))
```

(8) Extracting Digits of a Number Repeatedly

-Book by Charles Dierbach #5

Write a code to extract digits from a given number and print it. For example, If n=123 then it should print 1 2 3 12 23 123.

Code-

```
def s(n):
    a=[]
    for j in range(1,len(n)):
        for i in range(len(n)-j+1):
            a.append(n[i:i+j])
    a.append(n)
    print(*a)

n=int(input())

s(str(n))
```

(9) Factorial Number

--Book by Charles Dierbach #6

Write a code to find the factorial of a number by recursion.

Code-

```
def factorial(n):
   if n == 1:
      return n
   else:
      return n*factorial(n-1)
num=int(input("enter the number: "))
if num < 0:
   print("factorial can't be calculated")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   print("The factorial of ",num," is ", factorial(num))
```

(10) Traingle Quest 2

-HackerRank# 2

You are given a positive integer **N**.

Your task is to print a palindromic triangle of size **N**.

For example, a palindromic triangle of size 5 is:

1

121

12321

1234321

123454321

You can't take more than two lines. The first line (a *for*-statement) is already written for you.

You have to complete the code using exactly one print statement.

Note:

Using anything related to *strings* will give a score of 0 .

Using more than one *for*-statement will give a score of 0.

Input Format

A single line of input containing the integer N.

Constraints

- 0< N < 10

Output Format

Print the palindromic triangle of size N as explained above.

Sample Input

5

Sample Output

1
121
12321
1234321
123454321

CODE –

```
for i in range(1,int(input())+1):
 #More than 2 lines will result in 0 score. Do not leave a blank line a
lso
    print ((10**i//9)**2)
```

(11) Fibonacci Series

--Book by Charles Dierbach #7

Write a recursive code to find the Fibonacci series.

Code-

```
def fibonacci(n):
  if n<=1:
    return n
  else:
    return(fibonacci(n-1)+fibonacci(n-2))
num=int(input("How many terms you want:"))
for i in range(num):
   print(fibonacci(i)," ", end=" ")
print("...")
```

(12) Sum Of Taylor Expansion ($\sin x$)

--Book by Charles Dierbach #8

Write a code to find the sum of Taylor expansion of $\sin x$.

Code-

```
import math
def fact(k):
  if k<=1:
   return 1
  else:
   return k*fact(k-1)
step=int(input("How many terms : "))
x=int(input("Enter the value of x :"))
sum=0
for i in range(step+1):
   sum+=(math.pow(-1,i)*math.pow(x,2*i+1))/fact(2*i+1)
print("The result of sin",'(', x, ')', "is :", sum)
```

(13) Lucky Number

-Book by Charles Dierbach #9

Write a code to check whether number is lucky no. or not.

Lucky numbers are subset of integers. Let us see the process of arriving at lucky numbers,
Take the set of integers
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,……
First, delete every second number, we get following reduced set.
1,3,5,7,9,11,13,15,17,19,……
Now, delete every third number, we get
1, 3, 7, 9, 13, 15, 19,……..
Continue this process indefinitely……
Any number that does NOT get deleted due to above process is called “lucky”.
Therefore, set of lucky numbers is 1, 3, 7, 13,……

Code-

```
def isLucky(n):

    a = n

    if isLucky.counter > n:
        return 1
    if n % isLucky.counter == 0:
        return 0

        a = a - a /isLucky.counter

    isLucky.counter = isLucky.counter + 1

    return isLucky(next_position)



isLucky.counter = 2 x = 5
if isLucky(x):
    print (x,"is a Lucky number")
else:
    print (x,"is not a Lucky number")
```

(14) Greatest Common Divisor

--Book by Charles Dierbach #10

Write the greatest common divisor of two numbers recursively.

Code-

```
import sys

def gcd(p,q):
    if p<0 or q<0:
        sys.exit()
    if p==0:
        sys.exit()
    if q==0:
        return p
    else:
        return gcd(q,p%q)


p,q=int(input()),int(input())

print(gcd(p,q))
```

(15)Exceptions

-HackerRank# 3

Exceptions

Errors detected during execution are called *exceptions*.

Examples:

ZeroDivisionError

This error is raised when the second argument of a division or modulo operation

is zero.

```
>>> a = '1'
>>> b = '0'
>>> print int(a) / int(b)
>>> ZeroDivisionError: integer division or modulo by zero
```

ValueError

This error is raised when a built-in operation or function receives an argument

that has the right type but an inappropriate value.

```
>>> a = '1'
>>> b = '#'
>>> print int(a) / int(b)
>>> ValueError: invalid literal for int() with base 10: '#'
```

To learn more about different built-in exceptions click here.

Handling Exceptions

The statements *try* and *except* can be used to handle selected exceptions.

A *try* statement may have more than one except clause to specify handlers for

different exceptions.

```
#Code
try:
    print 1/0
except ZeroDivisionError as e:
    print "Error Code:",e
```

Output

```
Error Code: integer division or modulo by zero
```

Task

You are given two values a and b.

Perform integer division and print a/b.

Input Format

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

The next T lines each contain the space separated values of a and b.

Constraints

- 0<T<10

Output Format

Print the value of a/b.

In the case of *ZeroDivisionError* or *ValueError*, print the error code.

Sample Input

```
3
1 0
2 $
3 1
```

Sample Output

Error Code: integer division or modulo by zero
Error Code: invalid literal for int() with base 10: '$'
3

Note:

For integer divison in Python 3 use //.

Code –

```
x = int(input());
for i in range(x):
    try:
        a, b = input().split()
        print(int(a)//int(b))
    except ZeroDivisionError as e:
        print("Error Code:",e);
    except ValueError as v:
        print("Error Code:",v);
```

(16)Mod Divmod

-HackerRank# 4

One of the built-in functions of Python is *divmod*, which takes two arguments a and b returns a tuple containing the quotient of a/b first and then the remainder.

For example:

```
>>> print divmod(177,10)
(17, 7)
```

Here, the integer division is 177/10 => 17 and the modulo operator is 177%10 => 7.

Task

Read in two integers, a and b, and print three lines.

The first line is the integer division a//b (While using Python2 remember to import division from __future__).

The second line is the result of the modulo operator:a%b .

The third line prints the *divmod* of a and b.

Input Format

The first line contains the first integer,a , and the second line contains the second integer,b .

Output Format

Print the result as described above.

Sample Input

177
10
Sample Output

17
7
(17, 7)

Code –

```
a = int(input())
b = int(input())
print(a//b)
print(a%b)
print(divmod(a,b))
```

(17)Power - Mod Power

-HackerRank# 5

So far, we have only heard of Python's powers. Now, we will witness them!

Powers or exponents in Python can be calculated using the built-in power function. Call the power function a^b as shown below:

```
>>> pow(a,b)
```

or

```
>>> a**b
```

It's also possible to calculate a^b mod m.

```
>>> pow(a,b,m)
```

This is very helpful in computations where you have to print the resultant % mod.

Note: Here, a and b can be floats or negatives, but, if a third argument is present, b cannot be negative.

Note: Python has a math module that has its own *pow()*. It takes two arguments and returns a float. Frankly speaking, we will never use *math.pow()*.

Task

You are given three integers: a, b and m, respectively. Print two lines. The first line should print the result of *pow(a,b)*. The second line should print the result of *pow(a,b,m)*.

Input Format

The first line contains a, the second line contains b, and the third line contains .

Constraints

$1 \leq a \leq 10$

$1 \leq b \leq 10$

$2 \leq m \leq 1000$

Sample Input

```
3
4
5
```

Sample Output

```
81
1
```

Code –

```
a = int(input())
b = int(input())
m = int(input())

print(pow(a,b))

print(pow(a,b,m))
```

(18) Sum Of N series

You are given a number *n, n* being an integer. Write a python program to display the sum of the series up to *n* as given below:

Example

If *n* = 0, the series contains no terms and the sum is 0.

If *n* = 1, the series contain only 1 term which is: 1 and the sum is 1.

If *n* = 2, the series contain 2 terms: 2+22 and the sum is 24.

If *n* = 3, the series contain 3 terms: 3+33+333 and the sum is 369.

If *n* = 4, the series contain 4 terms: 4+44+444+4444 and the sum is 4936.

Take the value of '*n*' as input from the user. Print 'invalid input' if *n*<0.

Input Format

In the first line, take *n* as input

Output Format

Display the sum of the series

Sample Input 1

5

Sample Output 1

61725

Public:

5

61725

2

24

Private:

-1

Output:

invalid input

Private:

0

Output:

0

Code-

```
def power(x,i):

    if (i==0):

        return 1

    else:
        return x*power(x,i-1)


s=0
n=int(input())
if n<0:
    print('invalid input')
for I in range(n):

        for j in range(i+1):

            s+=n*power(10,j)


print(s)
```

(19) Sum of n numbers

Write a python program to display and find the sum of the series 1+11+111+....111 upto *n*. For example, If *n*=4, the series is: 1+11+111+1111. Take the value of '*n*' as input from the user.

Example

If *n* = 0, the series contains no terms and the sum is 0.

If *n* = 1, the series contain only 1 term which is: 1 and the sum is 1.

If *n* = 2, the series contain 2 terms: 1+11 and the sum is 12.

If *n* = 3, the series contain 3 terms: 1+11+111 and the sum is 123.

If *n* = 4, the series contain 4 terms: 1+11+111+1111 and the sum is 1234.

Take the value of '*n*' as input from the user. Print 'invalid input' if *n*<0.

Input Format

In the first line, take *n* as input

Output Format

Display the sum of the series

Sample Input 1

5

Sample Output 1

12345

Private:

-1

invalid input

Private:

0

0

Code-

```
def power(x,i):

    if (i==0):

        return 1

    else:
        return x*power(x,i-1)


s=0
n=int(input())
if n<0:
    print('invalid input')

for i in range(n):

        for j in range(i+1):

            s+=1*power(10,j)


print(s)
```

LOOPS (WHILE & FOR)

(20) While Loop

-Book by Charles Dierbach #11

Write a Python function named `getContinue` that displays to the user "`Do you want to continue(y/n): `", and continues to prompt the user until either uppercase or lowercase `'y'` or `'n'` is entered, returning (lowercase) `'y'` or `'n'` as the function value.

Code-

```
def getContinue():
    a=input("Do you want to continue (y/n):")

    while a!='y' or a!='n' or a!='Y' or a!='N':
        a=input("Do you want to continue (y/n):")
        if a=='y' or a=='n' or a=='Y' or a=='N':
            break
        else:
            continue

getContinue()
```

(21)Iterables and Iterators

-HackerRank# 6

The itertools module standardizes a core set of fast, memory efficient tools that are useful by themselves or in combination. Together, they form an iterator algebra making it possible to construct specialized tools succinctly and efficiently in pure Python.

You are given a list of N lowercase English letters. For a given integer K, you can select any K indices (assume 1-based indexing) with a uniform probability from the list.

Find the probability that at least one of the K indices selected will contain the letter: 'a'.

Input Format

The input consists of three lines. The first line contains the integer N , denoting the length of the list. The next line consists of N space-separated lowercase English letters, denoting the elements of the list.

The third and the last line of input contains the integer K, denoting the number of indices to be selected.

Output Format

Output a single line consisting of the *probability* that *at least* one of the K indices selected contains the letter: 'a'.

Note: The answer must be correct up to 3 decimal places.

Constraints

All the letters in the list are lowercase English letters.

Sample Input

4

a a c d

2

Sample Output

0.8333

Explanation

All possible unordered tuples of length 2 comprising of indices from 1 to 4 are:

(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)

Out of these 6 combinations, 5 of them contain either index 1 or index 2 which are the indices that contain the letter 'a'.

Hence, the answer is $\frac{5}{6}$.

Code –

```
from itertools import combinations
input()
combos = list(combinations(input().split(), int(input())))
count = 0
for i in combos:
    if "a" in i:
        count+=1
print(round(count/len(combos),3))
```

(22) Athlete Sort

-HackerRank# 7

You are given a spreadsheet that contains a list of N athletes and their details (such as age, height, weight and so on). You are required to sort the data based on the K^{th} attribute and print the final resulting table. Follow the example given below for better understanding.

Rank	Age	Height (in cm)		Rank	Age	Height (in cm)
1	32	190		5	24	176
2	35	175	sort based on k=1	4	26	195
3	41	188	⟶	1	32	190
4	26	195	i.e (age)	2	35	175
5	24	176		3	41	188

Note that K is indexed from 0 to M - 1, where M is the number of attributes.

Note: If two attributes are the same for different rows, for example, if two athletes are of the same age, print the row that appeared first in the input.

Input Format

The first line contains N and M separated by a space.
The next N lines each contain M elements.
The last line contains K.

Constraints

$1 \leq N, M \leq 1000$

$0 \leq K \leq M$

Each element ≤ 1000

Output Format

Print the N lines of the sorted table. Each line should contain the space separated elements. Check the sample below for clarity.

Sample Input 0

5 3

10 2 5

7 1 0

9 9 9

1 23 12

6 5 9

1

Sample Output 0

7 1 0

10 2 5

6 5 9

9 9 9

1 23 12

Explanation 0

The details are sorted based on the second attribute, since K is zero-indexed.

Code –

```
import math
import os
import random
import re
import sys

if __name__ == '__main__':
    nm = input().split()
    n = int(nm[0])
    m = int(nm[1])

    arr = []

    for _ in range(n):
        arr.append(list(map(int, input().rstrip().split())))
    k = int(input())


    arr.sort(key = lambda x : x[k])
    for i in arr:
        print(*i,sep=' ')
```

(23) Words Score

-HackerRank# 8

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

Consider that vowels in the alphabet are a, e, i, o, u and y.

Function score_words takes a list of lowercase words as an argument and returns a score as follows:

The score of a single word is 2 if the word contains an even number of vowels. Otherwise, the score of this word is 1. The score for the whole list of words is the sum of scores of all words in the list.

Debug the given function score_words such that it returns a correct score.

Your function will be tested on several cases by the locked template code.

Input Format

The input is read by the provided locked code template. In the first line, there is a single integer n denoting the number of words. In the second line, there are n space-separated lowercase words.

Constraints

- $1 \leq n \leq 20$
- Each word has at most 20 letters and all letters are English lowercase letters

Output Format

The output is produced by the provided and locked code template. It calls function score_words with the list of words read from the input as the argument and prints the returned score to the output.

Sample Input 0

```
2
hacker book
```

Sample Output 0

```
4
```

Explanation 0

There are two words in the input: hacker and book. The score of the word hacker is 2 because it contains an even number of vowels, i.e. 2 vowels, and the score of book is for the same reason. Thus the total score is 2+2=4.

Sample Input 1

```
3
programming is awesome
```

Sample Output 1

```
4
```

Explanation 1

There are 3 words in the input: programming, is and awesome. The score of programming is 1 since it contains 3 vowels, an odd number of vowels. The score of is also 1 because it has an odd number of vowels. The score of awesome is 2 since it contains vowels, an even number of vowels. Thus, the total score is 1+1+2=4.

Code –

```
import re
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']

def score_words(words):
    score = 0
    for word in words:
        num_vowels = 0
        for letter in word:
            if is_vowel(letter):
                num_vowels += 1
        if num_vowels % 2 == 0:
            score += 2
        else:
            score += 1
    return score



n = int(input())
words = input().split()
print(score_words(words))
```

STRINGS

(24) Two Strings

-HackerRank #9

Given two strings, determine if they share a common substring. A substring may be as small as one character.

Example

$s1 = \text{'and'}$
$s2 = \text{'art'}$

These share the common substring.

$s1 = \text{'be'}$
$s2 = \text{'cat'}$

These do not share a substring.

Function Description

Complete the function *twoStrings* in the editor below.

twoStrings has the following parameter(s):

- *string s1:* a string
- *string s2:* another string

Returns

- *string:* either YES or NO

Input Format

The first line contains a single integer, the number of test cases.

The following pairs of lines are as follows:

- The first line contains string.
- The second line contains string.

Constraints

- And consist of characters in the range ascii[a-z].
- 1<=p<=10
- $1<=|s1|,|s2|<=10^5$

Output Format

For each pair of strings, return YES or NO.

Sample Input

2
hello
world
hi
world

Sample Output

YES
NO

Explanation-

We have $p = 2$ pairs to check:

1. $s1 =$ `"hello"`, $s2 =$ `"world"`. The substrings `"o"` and `"l"` are common to both strings.
2. $a =$ `"hi"`, $b =$ `"world"`. $s1$ and $s2$ share no common substrings.

Code-

```
#!/bin/python3
import math
import os
import sys
# Complete the twoStrings function below.
def twoStrings(s1, s2):
    c=0
    for i in s1:
        if i in s2:
            c+=1
    if c>0:
        return ('YES')
    else:
        return ('NO')

if __name__ == '__main__':
    q = int(input())
    for q_itr in range(q):
        s1 = input()
        s2 = input()
        result = twoStrings(s1, s2)
        print(result)
```

(25) Exceptions - String to Integer

-HackerRank #10

Read a string, S, and print its integer value; if S cannot be converted to an integer, print Bad String.

Note: You *must* use the String-to-Integer and exception handling constructs built into your submission language. If you attempt to use loops/conditional statements, you will get a 0 score.

Input Format

A single string, S.

Constraints

- 1<=|S|<=6, where |S| is the length of string.
    - S is composed of *either* lowercase letters (a-z) *or* decimal digits (0-9).

Output Format

Print the parsed integer value of S, or Bad String if S cannot be converted to an integer.

Sample Input 0

3

Sample Output 0

3

Sample Input 1

za

Sample Output 1

Bad String

Explanation

Sample Case 0 contains an integer, so it should not raise an exception when we attempt to convert it to an integer. Thus, we print the 3.

Sample Case 1 does not contain any integers, so an attempt to convert it to an integer will raise an exception. Thus, our exception handler prints Bad String.

Code-

```
#!/bin/python3

s=input()
try:
    print(int(s))
except ValueError:
    print('Bad String')
```

(26) Sherlock and the Valid String

–HackerRank #11

Sherlock considers a string to be *valid* if all characters of the string appear the same number of times. It is also *valid* if he can remove just 1 character at 1 index in the string, and the remaining characters will occur the same number of times. Given a string, determine if it is *valid*. If so, return YES, otherwise return NO.

Example

s = abc

This is a valid string because frequencies are {a:1, b:1, c:1}.

s = abcc

This is a valid string because we can remove one c and have 1 of each character in the remaining string.

s = abccc

This string is not *valid* as we can only remove 1 occurrence of c. That leaves character frequencies of {a:1, b:1, c:2}.

Function Description

Complete the *isValid* function in the editor below.

isValid has the following parameter(s):

string s: a string

Returns

- *string:* either YES or NO

Input Format

A single string s.

Constraints

- $1<=|s|<=10^5$
- Each character s[i] belongs to ascii|a-z|

Sample Input 0

aabbcd

Sample Output 0

NO

Explanation 0

Given , s=" aabbcd" we would need to remove two characters,

both c and d → aabb or a and b → abcd, to make it valid. We are limited to

removing only one character, so s is *invalid*.

Sample Input 1

aabbccddeefghi

Sample Output 1

NO

Explanation 1

Frequency counts for the letters are as follows:

{'a': 2, 'b': 2, 'c': 2, 'd': 2, 'e': 2, 'f': 1, 'g': 1, 'h': 1, 'i': 1}

There are two ways to make the valid string:

- Remove 4 characters with a frequency of 1:{fghi}.
- Remove 5 characters of frequency 2:{abcde}.

Neither of these is an option.

Sample Input 2

abcdefghhgfedecba

Sample Output 2

YES

Explanation 2

All characters occur twice except for which occurs times. We can delete one instance of to have a valid string.

Code-

```
#!/bin/python3

import math
import os
import random
import re
import sys



# Complete the isValid function below.
def isValid(s):
    l=list(s)
    d={}
    for i in l:
        d[i]=l.count(i)
    print(d)
    maxi=0
    a=list(d.values())
    if len(a)==1:
        return 'YES'
    if len(a)==3 and min(a)==1:
        j=max(a)
        k=a.count(j)
        if k==2:
            return 'YES'
        else:
            return 'NO'
    for i in a:
        if a.count(i)>maxi:
            maxi=a.count(i)
    for i in a:
        if a.count(i)==maxi:
            e=a[i]
            break
    c=0
    for i in range(len(a)):
        if a[i]>e:
            a[i]=a[i]-1
            break
    try:
        for i in range(len(a)):
            if a[i]==a[i+1]:
                c+=1
```

```
    except:
        pass
    if c+1==len(a):
        return 'YES'
    else:
        return 'NO'




if __name__ == '__main__':
    s = input()
    result = isValid(s)
    print(result)
```

(27) Alternating Characters

-HackerRank #12

You are given a string containing characters A and B only. Your task is to change it into a string such that there are no matching adjacent characters. To do this, you are allowed to delete zero or more characters in the string.

Your task is to find the minimum number of required deletions.

Example

s=AABAAB

Remove an A at 0 positions and 3 to make s = ABAB in 2 deletions.

Function Description

Complete the *alternatingCharacters* function in the editor below.

alternatingCharacters has the following parameter(s):

- *string s*: a string

Returns

- *int:* the minimum number of deletions required

Input Format

The first line contains an integer q, the number of queries.

The next q lines each contain a string s to analyze.

Constraints

- 1<=q<=10
- 1<=length of s<=10^5
- Each string s will consist only of characters A and B.

Sample Input

5
AAAA
BBBBB
ABABABAB
BABABA
AAABBB

Sample Output

3
4
0
0
4

Explanation

The characters marked red are the ones that can be deleted so that the string does not have matching adjacent characters.

AAAA -> A (3 deletions)

BBBBB -> B (4 deletions)

ABABABAB -> ABABABAB (0 deletions)

BABABA -> BABABA (0 deletions)

AAABBB -> AB (4 deletions)

Code-

```
#!/bin/python3

import math
import os
import random
import re
import sys

# Complete the alternatingCharacters function below.
def alternatingCharacters(s):

    l,c=list(s),0
    #try:
    for i in range(len(l)-1):
            if l[i]==l[i+1]:
                #print(l)
                c+=1
                #l.remove(l[i+1])
                #print(l)
    '''except:
        pass'''


    return c

if __name__ == '__main__':
    q = int(input())

    for q_itr in range(q):
        s = input()

        result = alternatingCharacters(s)

        print(result)
```

(28) sWAP cASE

-HackerRank #13

You are given a string and your task is to *swap cases*. In other words, convert

all lowercase letters to uppercase letters and vice versa.

For Example:

Www.HackerRank.com → wWW.hACKERrANK.COM
Pythonist 2 → pYTHONIST 2

Input Format

A single line containing a string s.

Constraints

0<=len(s)<=1000

Output Format

Print the modified string s.

Sample Input 0

HackerRank.com presents "Pythonist 2".

Sample Output 0

hACKERrANK.COM PRESENTS "pYTHONIST 2".

Code-

```
def swap_case(result):
    return result.swapcase()



if __name__ == '__main__':
    s = input()
    result = swap_case(s)
    print(result)
```

(29) Text Wrap

-HackerRank #14

You are given a string S and width w.

Your task is to wrap the string into a paragraph of width w.

Input Format

The first line contains a string, S.

The second line contains the width, w.

Constraints

- 0<len(s)<1000
- 0<w<len(s)

Output Format

Print the text wrapped paragraph.

Sample Input 0

```
ABCDEFGHIJKLIMNOQRSTUVWXYZ
4
```

Sample Output 0

```
ABCD
EFGH
IJKL
IMNO
QRST
UVWX
YZ
```

Code-

```
import textwrap

def wrap(string, max_width):
    l=textwrap.wrap(string,width=max_width)
    return '\n'.join(l)


if __name__ == '__main__':
    string, max_width = input(), int(input())
    result = wrap(string, max_width)
    print(result)
```

(30) Palindrome String

--Book by Maureen Sprankle & Jim Hubbard #3

Write a solution to tell the user whether a string is a palindrome. (A palindrome string is a list of characters that spell the same word(s) forward or backward, such as *wow* or *radar.*)

Code-

```
def isStringPalindrome(str):
  if len(str)<=1:
    return True
  else:
    if str[0]==str[-1]:
      return isStringPalindrome(str[1:-1])
    else:
      return False
s=input("Enter the string : ")
y=isStringPalindrome(s)
if y==True:
   print("String is Palindrome")
else:
   print("String is Not Palindrome")
```

(31) String Validators

-HackerRank #15

Python has built-in string validation methods for basic data. It can check if a string is composed of alphabetical characters, alphanumeric characters, digits, etc.

str.isalnum()

This method checks if all the characters of a string are alphanumeric *(a-z, A-Z and 0-9)*.

```
>>> print 'ab123'.isalnum()
True
>>> print 'ab123#'.isalnum()
False
```

---

str.isalpha()

This method checks if all the characters of a string are alphabetical *(a-z and A-Z)*.

```
>>> print 'abcD'.isalpha()
True
>>> print 'abcd1'.isalpha()
False
```

---

str.isdigit()

This method checks if all the characters of a string are digits *(0-9)*.

```
>>> print '1234'.isdigit()
True
>>> print '123edsd'.isdigit()
False
```

---

str.islower()

This method checks if all the characters of a string are lowercase characters *(a-z)*.

```
>>> print 'abcd123#'.islower()
True
>>> print 'Abcd123#'.islower()
False
```

---

str.isupper()

This method checks if all the characters of a string are uppercase characters *(A-Z)*.

```
>>> print 'ABCD123#'.isupper()
True
>>> print 'Abcd123#'.isupper()
False
```

Task

You are given a string S.

Your task is to find out if the string S contains: *alphanumeric characters, alphabetical characters, digits, lowercase and uppercase characters*.

Input Format

A single line containing a string S.

Constraints

0<len(S)<1000

Output Format

In the first line, print True if S has any *alphanumeric characters*. Otherwise, print False.

In the second line, print True if S has any *alphabetical characters*. Otherwise, print False.

In the third line, print True if S has any *digits*. Otherwise, print False.

In the fourth line, print True if S has any *lowercase characters*. Otherwise, print False.

In the fifth line, print True if S has any *uppercase characters*. Otherwise, print False.

Sample Input

qA2

Sample Output

True
True
True
True
True

Code-

```
s=input()
an,a,d,l,u=0,0,0,0,0

for i in s:
    if i.isalnum():
        an+=1
    if i.isalpha():
        a+=1
    if i.isdigit():
        d+=1
    if i.islower():
        l+=1
    if i.isupper():
        u+=1

l=[an,a,d,l,u]

for i in l:
    if i>0:
        print(bool(1))
    else:
        print(bool(0))
```

(32) Mutations

-HackerRank #16

We have seen that lists are mutable (they can be changed), and tuples are

immutable (they cannot be changed).

Let's try to understand this with an example.

You are given an immutable string, and you want to make changes to it.

Example

```
>>> string = "abracadabra"
```
You can access an index by:

```
>>> print string[5]
a
```
What if you would like to assign a value?

```
>>> string[5] = 'k'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```
How would you approach this?

- One solution is to convert the string to a list and then change the value.

Example-

```
>>> string = "abracadabra"
>>> l = list(string)
>>> l[5] = 'k'
>>> string = ''.join(l)
>>> print string
abrackdabra
```
- Another approach is to slice the string and join it back.

Example-

```
>>> string = string[:5] + "k" + string[6:]
>>> print string
Abrackdabra
```

Task-

Read a given string, change the character at a given index and then print the modified string.

Input Format-

The first line contains a string, s.

The next line contains an integer i, denoting the index location and a character separated by a space.

Output Format-

Using any of the methods explained above, replace the character at index with character .

Sample Input-

```
abracadabra
5 k
```

Sample Output-

```
abrackdabra
```

Code-

```
def mutate_string(string, position, character):
    string=string[0:position]+character+s[position+1:]
    return string

if __name__ == '__main__':
    s = input()
    i, c = input().split()
    s_new = mutate_string(s, int(i), c)
    print(s_new)
```

(33) The Minion Game

-HackerRank #17

Kevin and Stuart want to play the '**The Minion Game**'.

Game Rules

Both players are given the same string, S.

Both players have to make substrings using the letters of the string S.

Stuart has to make words starting with *consonants*.

Kevin has to make words starting with *vowels*.

The game ends when both players have made all possible substrings.

Scoring

A player gets +1 point for each occurrence of the substring in the string S.

For Example:

String S= *BANANA*

Kevin's vowel beginning word = *ANA*

Here, *ANA* occurs twice in *BANANA*. Hence, Kevin will get 2 Points.

For better understanding, see the image below:

Your task is to determine the winner of the game and their score.

Input Format

A single line of input containing the string S.

Note: The string will contain only uppercase letters: |A-Z|.

Constraints

$1<=len(S)<=10^6$

Output Format

Print one line: the name of the winner and their score separated by a space.

If the game is a draw, print *Draw*.

Sample Input

BANANA

Sample Output

Stuart 12

Note:

Vowels are only defined as . In this problem, is not considered a vowel.

Code-

```
import string
S = str(raw_input())
L = len(S)

def score(S,L,letters):
    total = 0
    for index,char in enumerate(S) :
        if char in letters :
            points = L-index
            total+=points
            #print "index=",index,"char=",char,"points=",points,"total=",total
    return total

vowels = "AEIOU"
consonants = set(string.ascii_uppercase).difference(set(vowels))
score_1 = score(S,L,consonants)
#print "score 1 =",score_1
score_2 = score(S,L,vowels)
#print "score 2 =",score_2
if score_1 > score_2 :
    print("Stuart",score_1)
elif score_2 > score_1 :
    print("Kevin",score_2)
else :
    print("Draw")
```

(34)Company Logo

-HackerRank# 18

A newly opened multinational brand has decided to base their company logo on the three most common characters in the company name. They are now trying out various combinations of company names and logos based on this condition. Given a string s, which is the company name in lowercase letters, your task is to find the top three most common characters in the string.

- Print the three most common characters along with their occurrence count.
- Sort in descending order of occurrence count.
- If the occurrence count is the same, sort the characters in alphabetical order.

For example, according to the conditions described above,

GOOGLE would have it's logo with the letters G, O, E.

Input Format

A single line of input containing the string.

Constraints

- $3<\text{len}(S)\leq 10^4$

Output Format

Print the three most common characters along with their occurrence count each on a separate line. Sort output in descending order of occurrence count.

If the occurrence count is the same, sort the characters in alphabetical order.

Sample Input 0

aabbbccde

Sample Output 0

b 3
a 2
c 2

Explanation

aabbccde

Here, *b* occurs 3 times. It is printed first.

Both *a* and *c* occur 2 times. So, *a* is printed in the second line and *c* in the third line because *a* comes before *c* in the alphabet.

Note: The string S has at least 3 distinct characters.

Code –

```
import math
import os
import random
import re
import sys
from collections import Counter

class OrderedCounter(Counter):
    pass

if __name__ == '__main__':
    [print(*c) for c in OrderedCounter(sorted(input())).most_common(3)]
```

(35) ginortS

-HackerRank# 19

You are given a string S.

S contains alphanumeric characters only.

Your task is to sort the string S in the following manner:

- All sorted *lowercase letters* are ahead of *uppercase letters*.
- All sorted *uppercase letters* are ahead of digits.
- All sorted *odd digits* are ahead of sorted *even digits*.

Input Format

A single line of input contains the string S.

Constraints

- 0<len(S)<1000

Output Format

Output the sorted string S.

Sample Input

Sorting1234

Sample Output

ginortS1324

Code –

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
print(*sorted(input(), key=lambda c: (c.isdigit() -
 c.islower(), c in '02468', c)), sep='')
```

(36) Mutations

-HackerRank# 20

We have seen that lists are mutable (they can be changed), and tuples are immutable (they cannot be changed).

Let's try to understand this with an example.

You are given an immutable string, and you want to make changes to it.

Example

```
>>> string = "abracadabra"
```

You can access an index by:

```
>>> print string[5]
a
```

What if you would like to assign a value?

```
>>> string[5] = 'k'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

How would you approach this?

- One solution is to convert the string to a list and then change the value.

Example

```
>>> string = "abracadabra"
>>> l = list(string)
>>> l[5] = 'k'
>>> string = ''.join(l)
>>> print string
```

abrackdabra

- Another approach is to slice the string and join it back.

Example

```
>>> string = string[:5] + "k" + string[6:]
>>> print string
abrackdabra
```

Task
Read a given string, change the character at a given index and then print the modified string.

Input Format
The first line contains a string, S.
The next line contains an integer i, denoting the index location and a character c separated by a space.

Output Format
Using any of the methods explained above, replace the character at index i with character.

Sample Input

```
abracadabra
5 k
```

Sample Output

```
Abrackdabra
```

Code -

```
def mutate_string(string, position, character):
    l = list(string)
    l[position] = character;
    string = ''.join(l);
    return string

    return

if __name__ == '__main__':
    s = input()
    i, c = input().split()
    s_new = mutate_string(s, int(i), c)
    print(s_new)
```

(37) Text Alignment

-HackerRank# 21

In Python, a string of text can be aligned *left*, *right* and *center*.

.ljust(width)

This method returns a left aligned string of length *width*.

```
>>> width = 20
>>> print 'HackerRank'.ljust(width,'-')
HackerRank----------
```

.center(width)

This method returns a centered string of length *width*.

```
>>> width = 20
>>> print 'HackerRank'.center(width,'-')
-----HackerRank-----
```

.rjust(width)

This method returns a right aligned string of length *width*.

```
>>> width = 20
>>> print 'HackerRank'.rjust(width,'-')
----------HackerRank
```

Task

You are given a partial code that is used for generating the *HackerRank Logo* of

variable *thickness*.

Your task is to replace the blank (_____) with *rjust*, *ljust* or *center*.

Input Format

A single line containing the *thickness* value for the logo.

Constraints

0 < thickness < 50

The *thickness* must be an *odd* number.

Output Format

Output the desired logo.

Sample Input

5

Sample Output

```
    H
   HHH
  HHHHH
 HHHHHHH
HHHHHHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHHHHHHHHHHHHHHHHHHHHHH
  HHHHHHHHHHHHHHHHHHHHHHHHH
  HHHHHHHHHHHHHHHHHHHHHHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
  HHHHH          HHHHH
             HHHHHHHHHH
              HHHHHHH
```

HHHHH
HHH
H

Code –

```
thickness = int(input())
c = 'H'

#Top Cone
for i in range(thickness):
    print((c*i).rjust(thickness-1)+c+(c*i).ljust(thickness-1))

#Top Pillars
for i in range(thickness+1):
    print((c*thickness).center(thickness*2)+(c*thickness).center(thickn
ess*6))

#Middle Belt
for i in range((thickness+1)//2):
    print((c*thickness*5).center(thickness*6))

#Bottom Pillars
for i in range(thickness+1):
    print((c*thickness).center(thickness*2)+(c*thickness).center(thickn
ess*6))

#Bottom Cone
for i in range(thickness):
    print(((c*(thickness-i-1)).rjust(thickness)+c+(c*(thickness-i-
1)).ljust(thickness)).rjust(thickness*6))
```

(38)Text Wrap

-HackerRank# 22

You are given a string S and width w.

Your task is to wrap the string into a paragraph of width w.

Input Format

The first line contains a string, S.

The second line contains the width, w.

Constraints

- 0 < len(S) <1000
- 0<w<len(S)

Output Format

Print the text wrapped paragraph.

Sample Input 0

ABCDEFGHIJKLIMNOQRSTUVWXYZ
4

Sample Output 0

ABCD
EFGH
IJKL
IMNO
QRST
UVWX
YZ

Code –

```
import textwrap

def wrap(string, max_width):
    return textwrap.fill(string,max_width)


if __name__ == '__main__':
    string, max_width = input(), int(input())
    result = wrap(string, max_width)
    print(result)
```

(39)Designer Door Mate

-HackerRank# 23

Mr. Vincent works in a door mat manufacturing company. One day, he designed a new door mat with the following specifications:

- Mat size must be N X M. (N is an odd natural number, and M is times N .)
- The design should have 'WELCOME' written in the center.
- The design pattern should only use |, . and - characters.

Sample Designs

Size: 7 x 21

```
---------.|.---------
------.|..|..|.------
---.|..|..|..|..|.---
-------WELCOME-------
---.|..|..|..|..|.---
------.|..|..|.------
---------.|.---------
```

Size: 11 x 33

```
---------------.|.---------------
------------.|..|..|.------------
---------.|..|..|..|..|.---------
------.|..|..|..|..|..|..|.------
---.|..|..|..|..|..|..|..|..|.---
-------------WELCOME-------------
---.|..|..|..|..|..|..|..|..|.---
------.|..|..|..|..|..|..|.------
---------.|..|..|..|..|.---------
------------.|..|..|.------------
---------------.|.---------------
```

Input Format

A single line containing the space separated values of N and M .

Constraints

- 5 < N <101
- 15 < M<303

Output Format

Output the design pattern.

Sample Input

9 27

Sample Output

```
------------.|.------------
---------.|..|..|.---------
------.|..|..|..|..|.------
---.|..|..|..|..|..|..|.---
----------WELCOME----------
---.|..|..|..|..|..|..|.---
------.|..|..|..|..|.------
---------.|..|..|.---------
------------.|.------------
```

Code -

```
x,y = map(int,input().split())
items = list(range(1,x+1,2))
items = items+items[::-1][1:]
for i in items:
    text= "WELCOME" if i == x else '.|.'*i
    print (text.center(y,'-'))
```

(40)Alphabet Rangoli

-HackerRank# 24

You are given an integer, N. Your task is to print an alphabet rangoli of size N.

(Rangoli is a form of Indian folk art based on creation of patterns.)

Different sizes of alphabet rangoli are shown below:

#size 3

```
----c----
--c-b-c--
c-b-a-b-c
--c-b-c--
----c----
```

#size 5

```
--------e--------
------e-d-e------
----e-d-c-d-e----
--e-d-c-b-c-d-e--
e-d-c-b-a-b-c-d-e
--e-d-c-b-c-d-e--
----e-d-c-d-e----
------e-d-e------
--------e--------
```

#size 10

```
------------------j------------------
----------------j-i-j----------------
--------------j-i-h-i-j--------------
------------j-i-h-g-h-i-j------------
----------j-i-h-g-f-g-h-i-j----------
--------j-i-h-g-f-e-f-g-h-i-j--------
------j-i-h-g-f-e-d-e-f-g-h-i-j------
----j-i-h-g-f-e-d-c-d-e-f-g-h-i-j----
--j-i-h-g-f-e-d-c-b-c-d-e-f-g-h-i-j--
j-i-h-g-f-e-d-c-b-a-b-c-d-e-f-g-h-i-j
--j-i-h-g-f-e-d-c-b-c-d-e-f-g-h-i-j--
```

----j-i-h-g-f-e-d-c-d-e-f-g-h-i-j----
------j-i-h-g-f-e-d-e-f-g-h-i-j------
--------j-i-h-g-f-e-f-g-h-i-j--------
----------j-i-h-g-f-g-h-i-j----------
------------j-i-h-g-h-i-j------------
--------------j-i-h-i-j--------------
----------------j-i-j----------------
------------------j------------------

The center of the rangoli has the first alphabet letter *a*, and the boundary has the N^{th} alphabet letter (in alphabetical order).

Input Format

Only one line of input containing N, the size of the rangoli.

Constraints

$0 < N < 27$

Output Format

Print the alphabet rangoli in the format explained above.

Sample Input

5

Sample Output

--------e--------
------e-d-e------
----e-d-c-d-e----
--e-d-c-b-c-d-e--
e-d-c-b-a-b-c-d-e
--e-d-c-b-c-d-e--
----e-d-c-d-e----
------e-d-e------
--------e--------

Code –

```
def print_rangoli(size):
    import string
    alpha = string.ascii_lowercase

    L = []
    for i in range(n):
        s = "-".join(alpha[i:n])
        L.append((s[::-1]+s[1:]).center(4*n-3, "-"))

    print('\n'.join(L[:0:-1]+L))


if __name__ == '__main__':
    n = int(input())
    print_rangoli(n)
```

(41) Chandryaan-2

Ritika and John have developed a software containing python programs for the satellite application for "Chandrayaan-2" which will be invoked when it landed on the moon on September 7, 2019. They have stored this software in their computer machine which needs to be secured heavily. They would like to create a password for their machine. To create this, they came up with the following requirements:

A **strong password** is the one which:

- should be 10-12 characters long

- it should be alphanumeric(containing alphabets and digits). Alphanumeric means the presence of atleast 1 digit and 1 alphabet

- it should have atleast one uppercase and one lowercase letter

- it should have atleast one of the special characters from the given list: {@, $, !, %, #}

Weak Password:

- contains 6 – 9 characters

- contains alphanumeric characters(need not be both lower and upper case).

If it is a strong password, print valid and strong" or else "valid and weak". If the password does not fall into any of these categories, then print as "invalid".

Format:

In the first line get the password as input.

In the second line print whether it is valid or invalid.

Sample Input 1:

34ACX783bb

Sample Output 1:

valid and strong

- Be between 8-30 characters.
- Contain atleast one uppercase letter (A-Z)
- One lowercase letter (a-z)
- One number (0-9)
- One of the following special characters: !@#$%^&*+/*-.<>;'?[]{}

  Spaces are not allowed.

Code-

```
a=input()
b=len(a)
l,u,p,d=0,0,0,0
for i in a:
    if i.isupper():
        u+=1
    elif i.islower():
        l+=1
    elif i.isdigit():
        d+=1
    elif (i=='@') or (i=='$') or (i=='!') or (i=='%') or (i=='#'):
        p+=1
if (u>0) and (l>0) and (d>0) and (p>0) and (b>=10 and b<=12):
    print('valid and strong')
elif (u>0 ) and (l>0) and (p==0) and (b>=6 and b<=9):
    print('valid and weak')
else:
    print('invalid')
```

(42) Substring Ocurrence

You are given few sentences. You sequence of tasks is as follows:

First, remove all the special characters and change it to a full stop. This does not apply to already existing full stops.

Next, change all the uppercase letter with a lowercase letter.

After that, find the number of characters occurring contiguously for two times(e.g. rr, ll, mm) and print them along with their count of occurrences.

Input Format:

Take the sentences in English in a contiguous manner line by line.

Output Format:

In each line print the characters occurring contiguously for two times.

Print the total count of such occurrences.

Sample Input 1:

Surat - Based Sweet Shop Introduces 'Gold' Sweet @ Rs. 9,000/kg. As per a report in ANI, this sweet was launched ahead of Chandi Padvo, a Gujarati festival that falls a day after Sharad Purnima.

It will change to:

surat.based sweet shop introduces .gold. sweet . Rs. 9000.kg. as per a report in ani. this sweet was launched ahead of chandi padvo. a gujarati festival that falls a day after sharad purnima.

Sample Output 1:

ee
ee
ee
ll
4

My name is abc! I loove to play football@ I like progrraminG) They call me by aabc% My last NaMe is bbcd?

oo
oo
ll
rr
aa
bb
6

Code-

```
string=input()
string=list(string)
n=len(string)
count=0
for i in range(0,n):
    if (not string[i].isalpha()) and (not string[i].isnumeric()) and (not stri
ng[i]==' ') and (not string[i]=='.'):
        string[i]='.'
    string[i].lower()
    if (i<n-
1) and (not string[i].isnumeric()) and (not string[i]==' ') and (not string[i]
=='.'):
        if (string[i]==string[i+1]):
            print(string[i]*2)
            count+=1
string="".join(string)
print(count)
```

(43) Recursion- Subsequence

Write a recursive function that, given two strings/ integers/ alphanumeric, returns whether the first string is a
subsequence of the second. For example, please check the sample input and outputs. If the inputs contain a integer and a string, print as invalid.

Input Format:

In the first line, enter the first string/integer

In the second line, enter the second string/integer

Output Format:

In the first line, enter yes if the first string is a substring of the second else print no/invalid.

Sample Input 1:

hac

cathartic

Sample Output 1:

true

Sample Input 2:

2470

1234578

Sample Output 2:

false

private:

bat

table

false

40

12430

true

true

1234

invalid

Code-

```
def substring(c,d):
    if (len(c)==1) and (c[0] in d):
        print('true')
    elif c[0] in d:
        for i in range(0,len(d)):
            if c[0]==d[i]:
                substring(c[1:],d[i+1:])
    else:
        print('false')


a=input()
b=input()
if ((a.isnumeric() and b.isalpha()) or (b.isnumeric() and a.isalpha())):
    print('invalid')
else:
    substring(a,b)
```

(44) Isograms

An *isogram* (also known as a “nonpattern word”) is a logological term for a word or phrase in which no letter of the alphabet occurs more than once. A single letter word itself, is also an isogram.

Given a set of words, write a python program to display the shortest and the lengthiest ‘*isogram*’ of the list. Use a dictionary to store all the isogram/s along with its respective length. If there are no isograms, print ‘*none*’. If the words contain numerals/special characters, print ‘*invalid*’.

Input Format:

The first line contains the words separated by commas.

Output Format:

The first line prints the total number of isograms.

The second line prints the shortest isogram.

The third line prints the lengthiest isogram.

Examples:

Sample Input 1:

monkey,geek,python,is,bottle,best,ambidextrously

Sample Ouput 1:

5

is

ambidextrously

Sample Input 2:

ganga,94llman94,egmore,vellore

Sample Output 2:

none

private:

dermatoglyphic

output:

1

dermatoglyphic

dermatoglyphic

private:

1234,abc!@,study

Output:

invalid

Code-

```
import sys
from string import printable

def iso(l):
    d={}
    for I in l:
        if i.sidigit():
            print(“invalid”)
            sys.exit()

        if set(i).difference(printable):
            print(‘invalid’)
            sys.exit()
    for I in l:
        x=’’.join(dict.fromkeys(i))
        if len(i)==len(x):
            d[i]=len(i)
        else:
            continue

    if len(d)==0:
        print(‘none’)
    else:
        print(len(d))
        m,n=list(d.keys()),list(d.values())
        print(m[n.index(min(n))])
        print(m[n.index(max(n))])

l=list(map(str, input().split(‘,’)))
iso(l)
```

(45) Club Joining Eligibility

As a first year student, assume that you are interested to join in any of the club activities in your college to join the club, you are requested to feed your personal details in our college Club portal. The portal requires the following details: Register Number, First Name, School, Birthdate and Mobile Number. Write a python code to validate all the entries made by the student and print 'valid' or 'invalid' for each of the entries. Take care of the following conditions while you implement your program.

- A register number should be in this format: First two digits should be 20, followed by the school code which should be anyone from the *list*: {BPS, BCE, BAI, BRS, BEC, MEC}, followed by any four digits, E.g. 20BCE1234.

- Name should contain alphabets with first letter being an Uppercase.

- School name can be anyone from the following *list*: {SCOPE, SENSE,SMBS}.

- Birthdate should be of the format dd/mm/yyyy, where a date can be in the range 01-31, month in the range 01-12, year can be between 2000-2003.

- Mobile Number should contain exactly 10 digits with first digit not being 0.

Input Format:

In the first line, get the register number of the student
In the second line, get the first name
In the third line, get the school name
In the fourth line, get the date of birth
In the fifth line, get the mobile number

Output Format:

In the first line, print "valid" or "invalid" for the register number.
In the second line, print "valid" or "invalid" for the first name.
In the third line, print "valid" or "invalid" for the school name.
In the fourth line, print "valid" or "invalid" for the date of birth.
In the fifth line, print "valid" or "invalid" for the mobile number.

Sample Input 1:

20BAI1718
Garima
SCOPE
06/12/2001
1234567890

Sample Output 1:

valid
valid
valid
valid
valid

Sample Input 2:

20BAX1718
garima
SELECT
06/12/1999
9941567312

Sample Output 2:

invalid
invalid
invalid
invalid
valid

private:

20B!X1718
@Oswald
&@*#&
00/00/000x
98x561791!

Output-

invalid
invalid
invalid
invalid
invalid

private:

20B!X1718

&@*#&
00/00/000x
00000000!@

Output-

invalid
invalid
invalid
invalid
invalid

Code-

```
a=input()
b=input()
c=input()
d=input()
e=input()
a1,a2,a3,a4,a5=0,0,0,0,0

if a[0:2]==’20’ and a[2:5] in [‘BPS’,’BCE’,’BAI’,’BRS’,’BEC’,’MEC’] and a[5:].
isdigit():
    a1+=1

if b.isalpha() and b[0].isupper():
    a2+=1

if c in [‘SCOPE’,’SENSE’,’SMBS’]:
    a3+=1
if (int(d[0:2])>=1 and int(d[0:2])<=31) and (int(d[3:5])>=1 and int(d[3:5])<=1
2) and (int(d[6:])>=2000 and int(d[6:])<=2003):
    a4+=1


if int(e[0])!=0 and lenI==10 and e.isdigit():
    a5+=1


l=[a1,a2,a3,a4,a5]


for I in l:
    if i>0:
        print(‘valid’)
    else:
        print(‘invalid’)
```

REGEX

(46)Re.split()

-HackerRank# 25

You are given a string s consisting only of digits 0-9, commas, and dots.

Your task is to complete the regex_pattern defined below, which will be used to *re.split()* all of the and symbols in s.

It's guaranteed that every comma and every dot in s is preceeded and followed by a digit.

Sample Input 0

100,000,000.000

Sample Output 0

100
000
000
000

Code –

```
regex_pattern = r"[.,]" # Do not delete 'r'.

import re
print("\n".join(re.split(regex_pattern, input())))
```

(47)Re.findall() & Re.finditer()

-HackerRank# 26

The expression *re.findall()* returns all the non-overlapping matches of patterns in a string as a list of strings.

Code

```
>>> import re
>>> re.findall(r'\w','http://www.hackerrank.com/')
['h', 't', 't', 'p', 'w', 'w', 'w', 'h', 'a', 'c', 'k', 'e', 'r', 'r', 'a', 'n', 'k', 'c', 'o', 'm']
```

re.finditer()

The expression *re.finditer()* returns an iterator yielding MatchObject instances over all non-overlapping matches for the *re* pattern in the string.

Code

```
>>> import re
>>> re.finditer(r'\w','http://www.hackerrank.com/')
<callable-iterator object at 0x0266C790>
>>> map(lambda x: x.group(),re.finditer(r'\w','http://www.hackerrank.com/'))
['h', 't', 't', 'p', 'w', 'w', 'w', 'h', 'a', 'c', 'k', 'e', 'r', 'r', 'a', 'n', 'k', 'c', 'o', 'm']
```

Task

You are given a string S. It consists of alphanumeric characters, spaces and symbols(+,-).

Your task is to find all the substrings of S that contains 2 or more vowels.

Also, these substrings must lie in between 2 consonants and should contain vowels only.

Note :

Vowels are defined as: AEIOU **and** aeiou**.**

Consonants are defined

as: QWRTYPSDFGHJKLZXCVBNM **and** qwrtypsdfghjklzxcvbnm.

Input Format

A single line of input containing string S.

Constraints

0 < len(S) <100

Output Format

Print the matched substrings in their order of occurrence on separate lines.

If no match is found, print -1.

Sample Input

rabcdeefgyYhFjkIoomnpOeorteeeeet
Sample Output

ee
Ioo
Oeo
Eeeee

Explanation

ee is located between consonant d and f.

Ioo is located between consonant k and m .

Oeo is located between consonant p and r.

eeeee is located between consonant t and t.

Code –

```
import re

Storage = re.findall(r'(?<=[qwrtypsdfghjklzxcvbnm])([aeiou]{2,})(?=[qwr
typsdfghjklzxcvbnm])', input().strip(), re.IGNORECASE)

if Storage:
    for i in Storage:
        print(i)
else:
    print(-1)
```

(48)Validating phone numbers

-HackerRank# 27

Let's dive into the interesting topic of regular expressions! You are given some input, and you are required to check whether they are valid mobile numbers.

A valid mobile number is a ten digit number starting with 9 or 8.

Concept

A valid mobile number is a ten digit number starting with 9 or 8.

Regular expressions are a key concept in any programming language. A quick explanation with Python examples is available here. You could also go through the link below to read more about regular expressions in Python.

https://developers.google.com/edu/python/regular-expressions

Input Format

The first line contains an integer N, the number of inputs.

N lines follow, each containing some string.

Constraints

$1 \leq N \leq 10$

$2 \leq \text{len(Number)} \leq 15$

Output Format

For every string listed, print "*YES*" if it is a valid mobile number and "*NO*" if it is not on separate lines. Do not print the quotes.

Sample Input

2
9587456281
1252478965

Sample Output

YES
NO

Code –

```
import re

M = int(input())

for i in range(M):
    number = input()
    if(len(number)==10 and number.isdigit()):
        output = re.findall(r"^[789]\d{9}$",number)
        if(len(output)==1):
            print("YES")
        else:
            print("NO")
    else:
        print("NO")
```

(49)Validating Roman Numerals

-HackerRank# 28

You are given a string, and you have to validate whether it's a valid Roman numeral. If it is valid, print *True*. Otherwise, print *False*. Try to create a regular expression for a valid Roman numeral.

Input Format

A single line of input containing a string of Roman characters.

Output Format

Output a single line containing *True* or *False* according to the instructions above.

Constraints

The number will be between 1 and 3999 (both included).

Sample Input

CDXXI

Sample Output

True

References

Regular expressions are a key concept in any programming language. A quick explanation with Python examples is available here. You could also go through the link below to read more about regular expressions in Python.

https://developers.google.com/edu/python/regular-expressions

Code –

```
thousand = 'M{0,3}'
hundred = '(C[MD]|D?C{0,3})'
ten = '(X[CL]|L?X{0,3})'
digit = '(I[VX]|V?I{0,3})'
regex_pattern = r"%s%s%s%s$" % (thousand, hundred, ten, digit)
import re
print(str(bool(re.match(regex_pattern, input()))))
```

(50) Regex And Strings

In Indian official documents, the license number is listed as follows:

<alphabet>< alphabet> <digit><digit> < digit >< digit > <digit >< digit> <digit><digit><digit><digit><digit><digit><digit>

The first two alphabets represent the state then the two digits followed by a space, the next four digits represent the year followed by a space, the next 7 digits represent the serial number.

A valid license number will have all its characters in uppercase and digits in the same order as listed above. It should have a length of 17 along with white spaces.

The first two characters in the license number should only have any of the states from the following list:

KL, TN, MH, DL, OR, GJ, CH, BH, AP, TL, WB, HR, JK

The year should not start with 0 and can start with either the digit 1 or 2.

The first three digits of the serial number should be 0.

Your task is to figure out if a given license number is valid or not, with a python code. If it is valid print ‘valid’ else print ‘invalid’.

Sample Input:

First Line contains the License number

Sample Output:

Print valid or invalid

Sample Input 1:

TN11 2017 0006871

Sample Output 1:

Valid

Sample Input 2:

TN11 2017 0066871

Sample Output 2:

invalid

private:

ML11 2017 0006871

invalid

TN10 1997 0006871

Valid

Code-

```
a=input()
b=len(a)
c=['KL','TN','MH','DL','OR','GJ','CH','BH','AP','TL','WB','HR','JK']
for i in a:
    if (a[0:2] in c) and (a[2:4].isdigit()==True) and (a[4].isspace()==True) a
nd (a[5]=='1' or a[5]=='2') and (a[5:9].isdigit()==True) and (a[9].isspace()==
True) and (a[10:13]=='000') and (a[13:17].isdigit()==True):
        e='valid'
    else:
        e='invalid'
print(e)
```

(51)Incorrect Regex

-HackerRank# 29

You are given a string S.
Your task is to find out whether S is a valid regex or not.

Input Format

The first line contains integer T, the number of test cases.
The next T lines contains the string S.

Constraints

1< T <100

Output Format

Print "True" or "False" for each test case without quotes.

Sample Input

2

.*\+

.*+

Sample Output

True

False

Explanation

.*\+ : Valid regex.
.*+: Has the error multiple repeat. Hence, it is invalid.

Code –

```
import re;

N = int(input())
for _ in range(N):
    try:
        re.compile(input())
        Output = True
    except re.error:
        Output = False

    print(Output)
```

LISTS

(52) List of Fruits and Weights

-Book by Charles Dierbach #12

Write a Python program that prompts the user to enter types of fruit, and how many pounds of fruit there are for each type. The program should then display the information in the form *fruit* , *weight* listed in alphabetical order, one fruit type per line as shown below,

Apple, 6 lbs.

Banana, 11 lbs.

etc.

Code-

```
f=[]
w=[]
n=int(input("enter the total no. of fruits :"))
for i in range(n):
    a=input("enter fruit name: ")
    b=input("enter weight of fruit entered : ")
    f.append(a)
    w.append(b)

c=f.copy()
f.sort()

for i in range(len(f)):

    print(f[i],'-->',w[c.index(f[i])])
```

(53) List and Strings

--Book by Charles Dierbach #13

Write a Python program that prompts the user to enter a list of words and stores in a list only those words whose first letter occurs again within the word (for example, `'Baboon'`). The program should display the resulting list.

Code-

```
n=int(input("enter the total no. of words going to enter : "))
l=[]
for i in range(n):

    a=input("enter the words: ")

    if a[0] in a[1:]:

        l.append(a)

print("the accepted words are : ")
print(*l)
```

(54) Lowest Positive Element

--Book by Charles Dierbach #14

Given an array of integers, find the first missing positive integer in linear time and constant space. In other words, find the lowest positive integer that does not exist in the array. The array can contain duplicates and negative numbers as well. For example, the input `[3, 4, -1, 1]` should give `2`. The input `[1, 2, 0]` should give `3`.

You can modify the input array in-place.

Code-

```
def low_positive_element(l):

    maxi,mini=-999,999
    for i in l:
        if i>maxi:
            maxi=i
    for j in l:
        if j<mini:
            mini=j
    a=[]
    for i in range(mini,maxi):
        a.append(i)

    o=[]
    for i in a:
        if i not in l:
            o.append(i)

    if len(o)==0:
            print(maxi+1)
    else:
        if min(o)<0:
            o.remove(min(o))
            if len(o)==0:
                print(maxi+1)
            else:
                print(min(o))
        if 0 in o:
            o.remove(0)
            print(min(o))
        else:
            print(min(o))


l=list(map(int, input().split()))
low_positive_element(l)
```

(55) Arrays: Left Rotation

-HackerRank #30

A *left rotation* operation on an array shifts each of the array's elements 1 unit to the left. For example, 2 if left rotations are performed on array [1,2,3,4,5], then the array would become [3,4,5,1,2]. Note that the lowest index item moves to the highest index in a rotation. This is called a *circular array*.

Given an array a of n integers and a number, d perform d left rotations on the array. Return the updated array to be printed as a single line of space-separated integers.

Function Description

Complete the function *rotLeft* in the editor below.

rotLeft has the following parameter(s):

- *int a[n]:* the array to rotate
- *int d:* the number of rotations

Returns

- *int a'[n]:* the rotated array

Input Format

The first line contains two space-separated integers n and d, the size of a and the number of left rotations.

The second line contains n space-separated integers.

Constraints

- $1<=n<=10^5$
- $1<=d<=n$
- $1<=a[i]<=10^6$

Sample Input

5 4
1 2 3 4 5

Sample Output

5 1 2 3 4

Explanation

When we perform left rotations, the array undergoes the following sequence of changes:

$$[1,2,3,4,5] \rightarrow [2,3,4,5,1] \rightarrow [3,4,5,1,2] \rightarrow [4,5,1,2,3] \rightarrow [5,1,2,3,4]$$

Code-

```
#!/bin/python3

import math
import os
import random
import re
import sys

# Complete the rotLeft function below.
def rotLeft(a, d):
    b=[]
    #x=[]
    for i in range(len(a)):
        b.append(0)
    for i in range(len(a)):
             b[i-d]=a[i]
    return b

if __name__ == '__main__':
    nd = input().split()

    n = int(nd[0])

    d = int(nd[1])

    a = list(map(int, input().rstrip().split()))

    result = rotLeft(a, d)

    print(result)
```

(56) 2D Arrays

-HackerRank #31

Given a 6*6 *2D Array*, A:

1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

We define an hourglass in A to be a subset of values with indices falling in this pattern in A's graphical representation:

a b c
  d
e f g

There are 16 hourglasses in A, and an *hourglass sum* is the sum of an hourglass' values.

Task

Calculate the hourglass sum for every hourglass in A, then print the *maximum* hourglass sum.

Example

In the array shown above, the maximum hourglass sum is 7 for the hourglass in the top left corner.

Input Format

There are 6 lines of input, where each line contains 6 space-separated integers that describe the *2D Array A*.

Constraints

- $-9 \leq A[i][j] \leq 9$
- $0 \leq i, j \leq 5$

Output Format

Print the maximum hourglass sum in A.

Sample Input

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 2 4 4 0
0 0 0 2 0 0
0 0 1 2 4 0
```

Sample Output

```
19
```

Explanation

A contains the following hourglasses:

```
1 1 1   1 1 0   1 0 0   0 0 0
  1     0     0     0
1 1 1   1 1 0   1 0 0   0 0 0

0 1 0   1 0 0   0 0 0   0 0 0
  1     1     0     0
0 0 2   0 2 4   2 4 4   4 4 0

1 1 1   1 1 0   1 0 0   0 0 0
  0     2     4     4
0 0 0   0 0 2   0 2 0   2 0 0
```

0 0 2 0 2 4 2 4 4 4 4 0
 0   0   2   0
0 0 1 0 1 2 1 2 4 2 4 0

The hourglass with the maximum sum (19) is:

2 4 4
 2
1 2 4

Code-

```
a = [ [ int(i) for i in input().split() ] for _ in range(6) ]
print(max(sum(a[i][j:j+3] + [ a[i+1][j+1] ] + a[i+2][j:j+3]) for i in r
ange(4) for j in range(4)))
```

(57) Arrays

-HackerRank #32

Given an array, A, of N integers, print A's elements in *reverse* order as a single line of space-separated numbers.

Example-

A = [1,2,3,4]

Print 4 3 2 1. Each integer is separated by one space.

Input Format

The first line contains an integer, N (the size of our array).

The second line contains N space-separated integers that describe array A's elements.

Constraints

- $1 \leq N \leq 1000$
- $1 \leq A[i] \leq 10000$, where $A[i]$ is the i^{th} integer in the array.

Output Format

Print the elements of array A in reverse order as a single line of space-separated numbers.

Sample Input

4
1 4 3 2

Sample Output

2 3 4 1

Code-

```
n=int(input())
l=list(map(int, input().split()))
for i in range(1,len(l)+1):
    print(l[-i],end=' ')
```

(58) Last Stop Botique

-Book by Maureen Sprankle & Jim Hubbard #4

The Last Stop Boutique is having a five-day sale. Each day, starting on Monday, the price will drop 10% of the previous day's price. For example, if the original price of a product is $20.00, the sale price on Monday would be $18.00 (10% less than the original price). On Tuesday the sale price would be $16.20 (10% less than Monday). On Wednesday the sale price would be $14.58; on Thursday the sale price would be $13.12; and on Friday the sale price would be $11.81. Develop a solution that will calculate the price of an item for each of the five days, given the original price.

Code-

```
a=float(input("enter the actual price (in $): "))
l=[]
l.append(a-a*0.1)

for i in range(4):
    x=l[len(l)-1]-((l[len(l)-1])*0.1)
    l.append(x)

for i in range(5):
    print("price on ",i+1," day is ",format(l[i],'0.2f'))
```

(59) Recursive List Sum

--Book by Charles Dierbach #15

Write a recursive code to find the sum of list elements.

Code-

```
def s(l):

    if len(l)==1:
        return l[0]
    else:
        return m[len(l)-1]+s(l[0:len(l)-1])
        #return m[0]+s(m[1:])
m=[1,2,3,4,5]
print(s(m))
```

(60) Fraction Of Count

Given a list of integer values, find the fraction of count of positive numbers, negative numbers, zeroes to the total numbers and the sum of positive odd and even numbers. Print the value of the fractions correct to 2 decimal places.

Sample Input:

In the first line, take the number of integers ‘n’ in the list

In the second line, enter all the integers in the list

Sample Output:

Fraction of positive numbers in the list correct to 2 decimal places.

Fraction of negative numbers in the list correct to 2 decimal places.

Fraction of zeroes in the list correct to 2 decimal places.

Sum of positive odd and even numbers.

Sample Input 1:

10

4 5 -2 0 1 -56 9 0 2 0

Sample Output 1:

0.50

0.20

0.30

21

Private

5

0 0 0 0 0

0.00

0.00

1.00

0

Code-

```
n=int(input())
l=list(map(int, input().split()))
a=0
b=0
c=0
d=0
for I in l:
    if i>0:
        a+=1
print(format(a/n,'0.2f'))
for I in l:
    if i<0:
        b+=1
print(format(b/n,'0.2f'))
for I in l:
    if i==0:
        c+=1
print(format(c/n,'0.2f'))
for I in l:
    if i>0:
        d+=i
print(d)
```

(61)itertools.product()

-HackerRank# 33

itertools.product()

This tool computes the cartesian product of input iterables.

It is equivalent to nested *for-loops*.

For example,product(A, B) returns the same as ((x,y) for x in A for y in B).

Sample Code

```
>>> from itertools import product
>>>
>>> print list(product([1,2,3],repeat = 2))
[(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]
>>>
>>> print list(product([1,2,3],[3,4]))
[(1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4)]
>>>
>>> A = [[1,2,3],[3,4,5]]
>>> print list(product(*A))
[(1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 3), (3, 4), (3, 5)]
>>>
>>> B = [[1,2,3],[3,4,5],[7,8]]
>>> print list(product(*B))
[(1, 3, 7), (1, 3, 8), (1, 4, 7), (1, 4, 8), (1, 5, 7), (1, 5, 8), (2, 3, 7), (2, 3, 8), (2, 4, 7), (2, 4, 8), (2, 5, 7), (2, 5, 8), (3, 3, 7), (3, 3, 8), (3, 4, 7), (3, 4, 8), (3, 5, 7), (3, 5, 8)]
```

Task

You are given two lists A and B. Your task is to compute their cartesian

productA X B.

Example

A = [1, 2]
B = [3, 4]

AxB = [(1, 3), (1, 4), (2, 3), (2, 4)]
Note: A and B are sorted lists, and the cartesian product's tuples should be

output in sorted order.

Input Format

The first line contains the space separated elements of list A.

The second line contains the space separated elements of list B.

Both lists have no duplicate integer elements.

Constraints

0<A<30

0<B<30

Output Format

Output the space separated tuples of the cartesian product.

Sample Input

1 2
3 4

Sample Output

(1, 3) (1, 4) (2, 3) (2, 4)

Code –

```
from itertools import product
A = input().split()
A = list(map(int,A))
B = input().split()
B = list(map(int, B))
output = list(product(A,B))
for i in output:
    print(i, end = " ");
```

(62)The Captain's Room

-HackerRank# 34

Mr. Anant Asankhya is the manager at the *INFINITE* hotel. The hotel has an infinite amount of rooms.

One fine day, a *finite* number of tourists come to stay at the hotel.

The tourists consist of:

→ A Captain.

→ An unknown group of families consisting of K members per group where K ≠ 1.

The Captain was given a separate room, and the rest were given one room per group.

Mr. Anant has an unordered list of randomly arranged room entries. The list consists of the room numbers for all of the tourists. The room numbers will appear K times per group except for the Captain's room.

Mr. Anant needs you to help him find the Captain's room number.

The total number of tourists or the total number of groups of families is not known to you.

You only know the value of K and the room number list.

Input Format

The first line consists of an integer, N, the size of each group.

The second line contains the unordered elements of the room number list.

Constraints

$1 < K < 100$

Output Format

Output the Captain's room number.

Sample Input

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

Sample Output

```
8
```

Explanation

The list of room numbers contains elements. Since is, there must be groups of families. In the given list, all of the numbers repeat times except for room number.

Hence, is the Captain's room number.

Code –

```
N = int(input())

storage = map(int, input().split())
storage = sorted(storage)

for i in range(len(storage)):
    if(i != len(storage)-1):
        if(storage[i]!=storage[i-1] and storage[i]!=storage[i+1]):
            print(storage[i])
            break;
    else:
        print(storage[i])
```

(63)Collections.deque()

-HackerRank# 35

A *deque* is a double-ended queue. It can be used to add or remove elements from both ends.

Deques support thread safe, memory efficient appends and pops from either side of the deque with approximately the same 0(1) performance in either direction.

Click on the link to learn more about **deque() methods**.

Click on the link to learn more about various approaches to working with deques: **Deque Recipes**.

Example

Code

```
>>> from collections import deque
>>> d = deque()
>>> d.append(1)
>>> print d
deque([1])
>>> d.appendleft(2)
>>> print d
deque([2, 1])
>>> d.clear()
>>> print d
deque([])
>>> d.extend('1')
>>> print d
deque(['1'])
>>> d.extendleft('234')
>>> print d
deque(['4', '3', '2', '1'])
>>> d.count('1')
1
>>> d.pop()
```

```
'1'
>>> print d
deque(['4', '3', '2'])
>>> d.popleft()
'4'
>>> print d
deque(['3', '2'])
>>> d.extend('7896')
>>> print d
deque(['3', '2', '7', '8', '9', '6'])
>>> d.remove('2')
>>> print d
deque(['3', '7', '8', '9', '6'])
>>> d.reverse()
>>> print d
deque(['6', '9', '8', '7', '3'])
>>> d.rotate(3)
>>> print d
deque(['8', '7', '3', '6', '9'])
```

Task

Perform *append*, *pop*, *popleft* and *appendleft* methods on an empty deque d.

Input Format

The first line contains an integer N, the number of operations.

The next N lines contains the space separated names of methods and their

values.

Constraints

$0 < N \leq 100$

Output Format

Print the space separated elements of deque d.

Sample Input

6
append 1
append 2
append 3
appendleft 4
pop
popleft
Sample Output

1 2

Code –

```
from collections import deque

storage = deque()
N = int(input())

for i in range(N):
    io = input().split()
    if(io[0] == 'append'):
        storage.append(io[1])
    elif(io[0] == 'popleft'):
        storage.popleft()
    elif(io[0] == 'appendleft'):
        storage.appendleft(io[1])
    elif(io[0] == 'pop'):
        storage.pop()

print(' '.join(storage))
```

(64) Player Score

Write a program to create a tuple as given below for the sports player data. Find the total score of each player and print the score's percentage. Also, print maximum score from all players. Print 'invalid' if the given input is in wrong format.

Input Format:

In the first line, accept the number of players.

From second line onwards, get the player name and their five scores separated by spaces. For each player, get their details in a separate line.

Output Format:

In the first line, print the player's data as a tuple as given in the sample output format.

In the second line, print the player's score and percentage in a tuple embedded with his/her name in a dictionary. Finally, the dictionary entries should be enclosed as tuples. Please refer the sample output. The percentage value should have one digit after the decimal point.

In the third line, print the maximum score of all players.

Sample input 1:

2

preet 60 50 40 90 80

geet 50 60 40 80 99

Sample Output 1:

(('preet',60,50,40,90,80), ('geet',50,60,40,80,99))

({'preet': (320,64.0)}, {'geet': (329,65.8)})

329

Code-

```
from sys import exit
n=int(input())
l=[]
for i in range(n):
    x=input().split()
    a,b,c,d,e,f=x
    l.append((a,int(b),int(c),int(d),int(e),int(f)))

if n!=len(l):
    print('invalid')
    exit()
print(tuple(l))


d={}
big=-999
for i in range(len(l)):
    tot=l[i][1]+l[i][2]+l[i][3]+l[i][4]+l[i][5]
    if tot>big:
        big=tot
    d[l[i][0]]=(tot,format(tot/5,'.1f'))

print(d)

print(tot)
```

DICTIONARY

(65) Sales by Match

-HackerRank #36

There is a large pile of socks that must be paired by color. Given an array of integers representing the color of each sock, determine How many pairs of socks with matching colors there are?

Example:-

n=7

ar = [1,2,1,2,1,3,2]

There is one pair of color 1 and one of color 2. There are three odd socks left, one of each color. The number of pairs is 2.

Function Description

Complete the *sockMerchant* function in the editor below.

sockMerchant has the following parameter(s):

- *int n:* the number of socks in the pile
- *int ar[n]:* the colors of each sock

Returns

- *int:* the number of pairs

Input Format

The first line contains an integer, the number of socks represented in .

The second line contains space-separated integers, the colors of the socks in the pile.

Constraints

- 1<=n<=100
- 1<=ar[i]<=100 where 0<=i<n

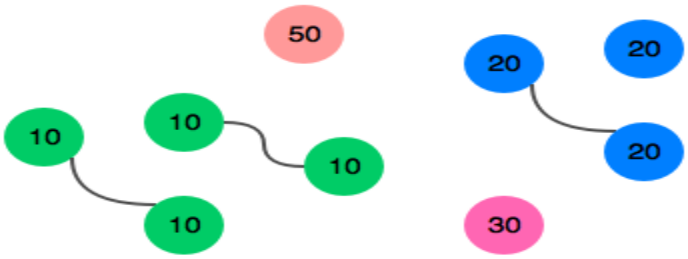
Sample Input

STDIN	Function
9	n = 9
10 20 20 10 10 30 50 10 20	ar = [10, 20, 20, 10, 10, 30, 50, 10, 20]

Sample Output

3

Explanation



There are three pairs of socks.

Code-

```
#!/bin/python3

import math
import os
import random
import re
import sys

# Complete the sockMerchant function below.
def sockMerchant(n, ar):
    d={}
    for i in ar:
        x=ar.count(i)
        d[i]=x
    c=0
    for i in list(d.values()):
        y=int(i/2)
        c+=y
    return c

if __name__ == '__main__':
     n = int(input())
    ar = list(map(int, input().rstrip().split()))
    result = sockMerchant(n, ar)
    print(result)
```

(66) Shop Dictionary

A new shop consists of a maximum of five different items like shoes, socks, belts, shiners, and bags. The items are in priority order that means shoes are of the highest priority and bags have the lowest priority. It maintains an inventory list manually which consists of the price of each item and the quantity (stock) of that item present in that shop. For example three items (shoes, socks, and belts) with 5 shoes each of 1000/-, 3 socks each of 100/- and 4 belts each of 300/-. This shoe company owner hires you to make an automated system such that your code could calculate the total stock and total cost of the inventory. It should also calculate the total cost of each item in the inventory. Design a python code using dictionaries to do this job.

Input Format:
In the first line, get the number of items.

For every items, get the following details in a new line

Item name 1
Cost of the item 1
Number of stocks of that item 1
.
.
.
Item name 'n'
Cost of the item 'n'
Number of stocks of that item 'n'

The input depends on the number of items entered by the user. Priority also matters. If the number of items is 3, then the first entry is shoes then socks and then belt. If the number of items is 4 then the first entry is shoes, then socks followed by the belt and finally shiner. The price and the stock entry is given in the example are just indicative.

Output Format:
For every item print the following in new lines.

Name of the item 1
Total cost of the item 1

.
.
.
Name of the item ‘n’
Total cost of the item ‘n’
Total stock of all the items
Total inventory cost of all the items correct to two decimal places

Sample Input 1:
2
shoes
1000
5
socks
100
3

Sample Output 1:
shoes
5000.00
socks
300.00
8
5300.00

Sample Input 2:
3
shoes
1000
5
socks
100
3
belts
500
2

Sample Output 2:
shoes
5000.00
socks
300.00
belts
1000.00
10
6300.00

Code-

```
n=int(input())
LIST={}
for i in range(0,n):
    name=input()
    price=float(input())
    number=int(input())
    LIST[i]=[name,price,number]
SUM=0
sumnum=0
for i in range(0,n):
    print(LIST[i][0])
    tempsum=(float(LIST[i][1]))*(int(LIST[i][2]))
    print(format(tempsum,'.2f'))
    SUM=SUM+tempsum
    sumnum=sumnum+LIST[i][2]

print(sumnum)
print(format(SUM,'.2f')
```

(67)DefaultDict Tutorial

-HackerRank# 37

The *defaultdict* tool is a container in the collections class of Python. It's similar to the usual dictionary (*dict*) container, but the only difference is that a defaultdict will have a *default* value if that key has not been set yet. If you didn't use a defaultdict you'd have to check to see if that key exists, and if it doesn't, set it to what you want.

For example:

```
from collections import defaultdict
d = defaultdict(list)
d['python'].append("awesome")
d['something-else'].append("not relevant")
d['python'].append("language")
for i in d.items():
    print i
```

This prints:

```
('python', ['awesome', 'language'])
('something-else', ['not relevant'])
```

In this challenge, you will be given 2 integers, n and m. There are n words, which might repeat, in word group A. There are m words belonging to word group B. For each m words, check whether the word has appeared in group A or not. Print the indices of each occurrence of in group A . If it does not appear, print -1.

Constraints

$1 \leq n \leq 10000$

$1 \leq m \leq 100$

$1 \leq$ length of each word in the input ≤ 100

Input Format

The first line contains n and m integers, separated by a space.

The next n lines contains the words belonging to group .

The next m lines contains the words belonging to group .

Output Format

Output m lines.

The line should contain the -indexed positions of the occurrences of the word

separated by spaces.

Sample Input

```
5 2
a
a
b
a
b
a
b
```

Sample Output

1 2 4
3 5

Explanation

'**a**' appeared times in positions 1,2 and 4.

'**b**' appeared times in positions 3 and 5.

In the sample problem, if '**c**' also appeared in word group B, you would print -

1 .

Code –

```
from collections import defaultdict

n,m = list(map(int,input().split()))
d = defaultdict(list)
for i in range(n):
    d[input()].append(i+1)
for i in range(m):
    print(*d[input()] or [-1])
```

(68)Word Order

-HackerRank# 38

You are given n words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.

Note: Each input line ends with a **"\n"** character.

Constraints:

$1 \leq n \leq 10^5$

The sum of the lengths of all the words do not exceed 10^6

All the words are composed of lowercase English letters only.

Input Format

The first line contains the integer, n .

The next n lines each contain a word.

Output Format

Output 2 lines.

On the first line, output the number of distinct words from the input.

On the second line, output the number of occurrences for each distinct word according to their appearance in the input.

Sample Input

4
bcdef
abcdefg
bcde
bcdef

Sample Output

3
2 1 1

Explanation

There are 3 distinct words. Here, "**bcdef**" appears twice in the input at the first and last positions. The other words appear once each. The order of the first appearances are "**bcdef**", "**abcdefg**" and "**bcde**" which corresponds to the output.

Code –

```
import collections;

N = int(input())
d = collections.OrderedDict()

for i in range(N):
    word = input()
    if word in d:
        d[word] +=1
    else:
        d[word] = 1

print(len(d));

for k,v in d.items():
    print(v,end = " ");
```

(69) Sentence Correction

Abhijit Banerjee, the Indian born American is the Nobel prize winner for the year 2019 in Economics. He retrieved a text from the web for his research work. Unfortunately, he was surprised to see there were many duplicate words present in the text. He is in need of a text which contains only unique words. Help Abhijit to write a python code to achieve this task without using inbuilt methods. If there are no duplicates, print the text as it is.

Input Format:

The first line contains the text. Assume there are no special characters in the text.

Output Format:

In the first line, print the duplicate word and the number of times it occurs in the text, if any.

In the next line, print the indices of the duplicate words in the text, with the first index being 0.

Repeat the above two steps for other duplicate words, if any.

In the last line, print the updated text containing unique words.

Sample Input 1:

python abhijit turing 154llman python linus turing deitel turing

Sample Output 1:

python 2

0 4

turing 3

2 6 8

python abhijit turing 154llman linus deitel

Sample Input 2:

python abhijit turing 155llman linus deitel

Sample Output 2:

python abhijit turing 155llman linus deitel

---

private 1:

simputer jeff1 895 boole stroutstrup jeff1

Output 1-

jeff1 2

1 5

simputer jeff1 895 boole stroutstrup

private 2:

hello123

Output 2-

hello123

Code-

```
import sys
b=input()
a=b.split()

if b == ' '.join(dict.fromkeys(a)):
    print(b)
    sys.exit()
d={}

for I in a:
    d[i]=a.count(i)

x,y=list(d.keys()),list(d.values())

for I in range(len(y)):
    if y[i]>1:
        print(x[i],y[i])
        o=[]
        for j in range(len(a)):
            if x[i]==a[j]:
                o.append(j)
        print(*o)

print(' '.join(dict.fromkeys(a)))
```

SETS

(70) No Idea!

-HackerRank #39

There is an array of n integers. There are also 2 **disjoint sets**, A and B, each containing m integers. You like all the integers in set A and dislike all the integers in set B. Your initial happiness 0 is i. For each integer in the array, if i belongs to A, you add 1 to your happiness. If i belongs to B, you add -1 to your happiness. Otherwise, your happiness does not change. Output your final happiness at the end.

Note: Since A and B are sets, they have no repeated elements. However, the array might contain duplicate elements.

Constraints

$1<=n<=10^5$

$1<=m<=10^5$

$1<=$any integer in the input$<=10^8$

Input Format

The first line contains two integers n and m separated by a space.
The second line contains n integers, the elements of the array.
The third and fourth lines contain m integers, A and B, respectively.

Output Format

Output a single integer, your total happiness.

Sample Input

3 2

1 5 3

3 1

5 7

Sample Output

1

Explanation

You gain 1 unit of happiness for elements 3 and 1 in set A. You lose 1 unit for 5 in set B. The element 7 in set B does not exist in the array so it is not included in the calculation.

Hence, the total happiness is 2-1=1.

Code-

```
a=list(map(int, input().split()))
b=list(map(int, input().split()))
c=set(map(int, input().split()))
d=set(map(int, input().split()))

i=0

for j in b:
    if j in c:
        i+=1
    if j in d:
        i-=1
    else:
        i+=0


print(i)
```

(71) Set .discard(), .remove() & .pop()

-HackerRank #40

.remove(x)

This operation removes element x from the set.

If element x does not exist, it raises a KeyError.

The *.remove(x)* operation returns None.

Example

```
>>> s = set([1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> s.remove(5)
>>> print s
set([1, 2, 3, 4, 6, 7, 8, 9])
>>> print s.remove(4)
None
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
>>> s.remove(0)
KeyError: 0
```

.discard(x)

This operation also removes element x from the set.

If element x does not exist, it **does not** raise a KeyError.

The *.discard(x)* operation returns None.

Example

```
>>> s = set([1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> s.discard(5)
>>> print s
set([1, 2, 3, 4, 6, 7, 8, 9])
>>> print s.discard(4)
None
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
>>> s.discard(0)
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
```

.pop()

This operation removes and return an arbitrary element from the set.

If there are no elements to remove, it raises a `KeyError`.

Example

```
>>> s = set([1])
>>> print s.pop()
1
>>> print s
set([])
>>> print s.pop()
KeyError: pop from an empty set
```

Task

You have a non-empty set S, and you have to execute N commands given in N lines.

The commands will be *pop*, *remove* and *discard*.

Input Format

The first line contains integer n, the number of elements in the set s.

The second line contains n space separated elements of set s. All of the elements are non-negative integers, less than or equal to 9.

The third line contains integer N, the number of commands.

The next N lines contains either *pop*, *remove* and/or *discard* commands followed by their associated value.

Constraints

0<n<20

0<N<20

Output Format

Print the sum of the elements of set s on a single line.

Sample Input

```
9
1 2 3 4 5 6 7 8 9
10
pop
remove 9
discard 9
discard 8
remove 7
pop
discard 6
remove 5
pop
discard 5
```

Sample Output

```
4
```

Explanation

After completing these 10 operations on the set, we get *set(|4|)*. Hence, the sum is 4.

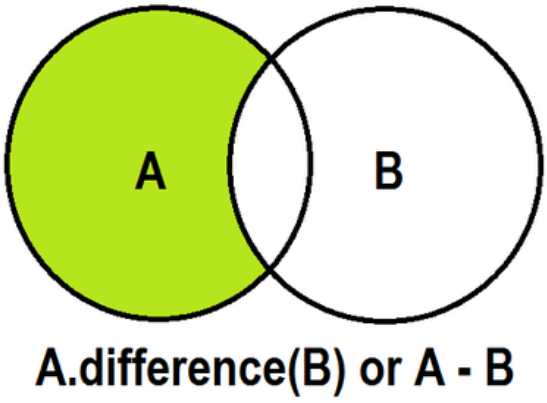
Note: Convert the elements of set *s* to *integers* while you are assigning them. To ensure the proper input of the set, we have added the first two lines of code to the editor.

Code-

```
import sys
n = int(input())
s = set(map(int, input().split()))
for i in s:
    if i<0 or i>9:
        sys.exit()
N=int(input())
for i in range(N):
    a=list(map(str, input().split()))
    if a[0]=='pop':
        s.pop()
    if a[0]=='remove':
        s.remove(int(a[1]))
    if a[0]=='discard':
        s.discard(int(a[1]))

print(sum(s))
```

(72)Set .difference() Operation

-HackerRank# 41



By DOSHI

.difference()

The tool *.difference()* returns a set with all the elements from the set that are not in an iterable.

Sometimes the - operator is used in place of the *.difference()* tool, but it only operates on the set of elements in *set*.

Set is immutable to the *.difference()* operation (or the - operation).

```
>>> s = set("Hacker")
>>> print s.difference("Rank")
set(['c', 'r', 'e', 'H'])

>>> print s.difference(set(['R', 'a', 'n', 'k']))
set(['c', 'r', 'e', 'H'])

>>> print s.difference(['R', 'a', 'n', 'k'])
set(['c', 'r', 'e', 'H'])

>>> print s.difference(enumerate(['R', 'a', 'n', 'k']))
```

```
set(['a', 'c', 'r', 'e', 'H', 'k'])

>>> print s.difference({"Rank":1})
set(['a', 'c', 'e', 'H', 'k', 'r'])

>>> s - set("Rank")
set(['H', 'c', 'r', 'e'])
```

Task

Students of District College have a subscription to *English* and *French* newspapers. Some students have subscribed to only the *English* newspaper, some have subscribed to only the *French* newspaper, and some have subscribed to both newspapers.

You are given two sets of student roll numbers. One set has subscribed to the *English* newspaper, and one set has subscribed to the *French* newspaper. Your task is to find the total number of students who have subscribed to *only English* newspapers.

Input Format

The first line contains the number of students who have subscribed to the *English* newspaper.

The second line contains the space separated list of student roll numbers who have subscribed to the *English* newspaper.

The third line contains the number of students who have subscribed to the *French* newspaper.

The fourth line contains the space separated list of student roll numbers who have subscribed to the *French* newspaper.

Constraints

0 < Total number of student in college <1000

Output Format

Output the total number of students who are subscribed to the *English* newspaper *only*.

Sample Input

```
9
1 2 3 4 5 6 7 8 9
9
10 1 2 3 11 21 55 6 8
```

Sample Output

4

Explanation

The roll numbers of students who *only* have *English* newspaper subscriptions are:

4,5,7 and 9 .

Hence, the total is 4 students.

Code –

```
N1 = int(input())
storage1 = set(input().split())

N2 = int(input())
storage2 = set(input().split())

storage3 = storage1.difference(storage2)

print(len(storage3))
```

(73) Sets

A super market plans to conduct a surprise cum lucky winner contest as an offer for the upcoming Diwali. They decided to give a prize to two customers with the maximum number of items in common in their shopping list. Write a program to identify common items from the shopping list of two customers and print the winners and the count of the common items. Prepare the common items list as read only. If there are more than 2 customers with the maximum common items, print the first two customers with respect to their customer id. If no winners are there, print as 'no winners'.

Input Format:

In the first line get *n,* which is the number of customers so far purchased.

In the second line, get the items bought by the customer with a customer *id* for every customer, separated by a space and the set of items as their product id (Separated by spaces). A product id can't be 0 or negative. If so, print as invalid input.

Output Format:

In the first line, print the winners with the customer id's separated by space

In the second line, print their common items as product id's in their basket.

In the third line, print the number of common items ordered by their product id

Sample Input 1:

3

1 1 4 2 5 6

2 2 4 3 1 6 7

3 1 2 4 6 5

Sample Output 1:

1 3

1 2 4 5 6

5

Sample Input 2:

5

1 3 8 2 90 33 4

2 90 4

3 3 90 33 6 4 88

4 7 9 1 2 5

5 7 2 33 90 100 4 3

Sample Output 2:

1 5

2 3 4 33 90

5

--

3

1 4 2

2 3 4 5

3 0 0 0

invalid input

2

1 4 2 1 3

2 8 9 6 5

no winners

Code-

```
n=int(input())
flag=1
products={}
costumers=[]
for i in range(0,n):
    List=input()
    List=List.split()
    m=len(List)
    for j in range(0,m):
        List[j]=int(List[j])
        if List[j]<=0:
            print("invalid input")
            exit()
    products.update({List[0]:set(List[1:])})
    costumers.append(List[0])
Max=0
costumers=sorted(costumers,reverse=True)
for i in costumers:
    for j in costumers:
        if i!=j:
            if len(products[i]&products[j])>Max:
                w1=i
                w2=j
                Max=len(products[i]&products[j])
                flag=0

if flag==0:
    print(w2,w1)
    final=sorted(list(products[w1]&products[w2]))
    print(" ".join(str(i) for i in final))
    print(len(final))
else:
    print("no winners")
```

(74)Check Subset

-HackerRank# 42

You are given two sets, A and B.

Your job is to find whether set A is a subset of set B.

If set A is subset of set B, print **True**.

If set A is not a subset of set B, print **False**.

Input Format

The first line will contain the number of test cases, T.

The first line of each test case contains the number of elements in set A.

The second line of each test case contains the space separated elements of set A.

The third line of each test case contains the number of elements in set B.

The fourth line of each test case contains the space separated elements of set B.

Constraints

- 0<T <21
- 0<Number of elements in each set < 1001

Output Format

Output **True** or **False** for each test case on separate lines.

Sample Input

```
3
5
1 2 3 5 6
9
9 8 5 6 3 2 1 4 7
1
2
5
3 6 5 4 1
7
1 2 3 5 6 8 9
3
9 8 2
```

Sample Output

```
True
False
False
```

Explanation

Test Case 01 Explanation

Set A = {1 2 3 5 6}

Set B= {9 8 5 6 3 2 1 4 7}

All the elements of set A are elements of set B.

Hence, set A is a subset of set B.

Code –

```
T = int(input())

for _ in range(T):
    a = input()
    A = set(input().split())
    b = int(input())
    B = set(input().split())
    print(A.issubset(B))
```

FUNCTIONS & CLASSES

(75) Write a function

-HackerRank #43

An extra day is added to the calendar almost every four years as February 29, and the day is called a *leap day*. It corrects the calendar for the fact that our planet takes approximately 365.25 days to orbit the sun. A leap year contains a leap day.

In the Gregorian calendar, three conditions are used to identify leap years:

- The year can be evenly divided by 4, is a leap year, unless:
    - The year can be evenly divided by 100, it is NOT a leap year, unless:
        - The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years. Source

Task

Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean `True`, otherwise return `False`.

Note that the code stub provided reads from STDIN and passes arguments to the `is_leap` function. It is only necessary to complete the `is_leap` function.

Input Format

Read year, the year to test.

Constraints

1990<year<10^5

Output Format

The function must return a Boolean value (True/False). Output is handled by the provided code stub.

Sample Input 0

1990

Sample Output 0

False

Explanation 0

1990 is not a multiple of 4 hence it's not a leap year.

Code-

```
def is_leap(year):
    leap=False
    if year%4==0:
        leap=True
    if year%100==0:
        leap=False
    if year%400==0:
        leap=True
    return leap

year = int(input())
print(is_leap(year))
```

(76) Scope

-HackerRank #44

The *absolute difference* between two integers, a and b, is written as |a-b| .

The *maximum absolute difference* between two integers in a set of positive integers, elements, is the largest absolute difference between any two integers in _elements.

The *Difference* class is started for you in the editor. It has a private integer array (element) for storing N non-negative integers, and a public integer (maximumDifference) for storing the maximum absolute difference.

Task

Complete the *Difference* class by writing the following:

- A class constructor that takes an array of integers as a parameter and saves it to the _elements, instance variable.
- A *computeDifference* method that finds the maximum absolute difference between any 2 numbers in _elements and stores it in the maximumDifference instance variable.

Input Format

You are not responsible for reading any input from stdin. The locked *Solution* class in the editor reads in lines of input. The first line contains, the size of the elements array. The second line has space-separated integers that describe the array.

Constraints

- 1<=N<=10
- 1<=_elements[i]<=100, where 0<=i<=N-1

Output Format

You are not responsible for printing any output; the *Solution* class will print the value of the maximumDifference instance variable.

Sample Input

STDIN	Function
3	__elements[] size N = 3
1 2 5	__elements = [1, 2, 5]

Sample Output

4

Explanation

The scope of the _elements array and maximumDifference integer is the entire class instance. The class constructor saves the argument passed to the

constructor as the maximumDifference instance variable (where the *computeDifference* method can access it).

To find the maximum difference, *computeDifference* checks each element in the array and finds the maximum difference between any 2 elements: |1-2|=1

|1-5|=4

|2-5|=3

The maximum of these differences is 4, so it saves the value 4 as the maximumDifference instance variable. The locked stub code in the editor then prints the value stored as maximumDifference, which is 4.

Code-

```
class Difference:
    def __init__(self, a):
        self.__elements = a
    def computeDifference(self):
        self.maximumDifference = 0
        for x in range(len(a)):
            for y in range(x,len(a)):
                if abs(a[x] - a[y]) > self.maximumDifference:
                    self.maximumDifference = abs(a[x] - a[y])

# End of Difference class

_ = input()
a = [int(e) for e in input().split(' ')]

d = Difference(a)
d.computeDifference()

print(d.maximumDifference)
```

(77) Abstract Classes

-HackerRank #45

Given a *Book* class and a *Solution* class, write a *MyBook* class that does the following:

- Inherits from *Book*
- Has a parameterized constructor taking these 3 parameters:
  1. string title
  2. string author
  3. int price
- Implements the *Book* class' abstract *display()* method so it prints these 3 lines:
  1. Title, a space, and then the current instance's title.
  2. Author, a space, and then the current instance's author.
  3. Price, a space, and then the current instance's price.

Note: Because these classes are being written in the same file, you must not use an access modifier (e.g.: public) when declaring *MyBook* or your code will not execute

Input Format

You are not responsible for reading any input from stdin. The *Solution* class creates a *Book* object and calls the *MyBook* class constructor (passing it the necessary arguments). It then calls the *display* method on the *Book* object.

Output Format

The void display() method should print and label the respective title, author and price of the *MyBook* object's instance (with each value on its own line) like so:

Title: $title
Author: $author
Price: $price

Note: The $ is prepended to variable names to indicate they are placeholders for variables.

Sample Input

The following input from stdin is handled by the locked stub code in your editor:

The Alchemist
Paulo Coelho
248

Sample Output

The following output is printed by your *display()* method:

Title: The Alchemist
Author: Paulo Coelho
Price: 248

Code-

```
from abc import ABCMeta, abstractmethod
class Book(object, metaclass=ABCMeta):
    def __init__(self,title,author):
        self.title=title
        self.author=author
    @abstractmethod
    def display(): pass
#Write MyBook class
class MyBook(Book):
    def __init__(self,title,author,price):
        Book.__init__(self,title,author)
        self.price = price
    def display(self):
        print("Title: " + self.title)
        print("Author: " + self.author)
        print("Price: " + str(self.price))



title=input()
author=input()
price=int(input())
new_novel=MyBook(title,author,price)
new_novel.display()
```

(78) Inheritance

-HackerRank #46

You are given two classes, *Person* and *Student*, where *Person* is the base class and *Student* is the derived class. Completed code for *Person* and a declaration for *Student* are provided for you in the editor. Observe that *Student* inherits all the properties of *Person*.

Complete the *Student* class by writing the following:

- A *Student* class constructor, which has parameters:
  1. A string, firstName.
  2. A string, lastName.
  3. An integer, idNumber.
  4. An integer array (or vector) of test scores, scores.
- A *char calculate()* method that calculates a Student object's average and returns the grade character representative of their calculated average:

Grading Scale

Letter	Average (a)
O	$90 \le a \le 100$
E	$80 \le a < 90$
A	$70 \le a < 80$
P	$55 \le a < 70$
D	$40 \le a < 55$
T	$a < 40$

Input Format

The locked stub code in the editor reads the input and calls the *Student* class constructor with the necessary arguments. It also calls the *calculate* method which takes no arguments.

The first line contains, firstName, lastName and idNumber, separated by a space. The second line contains the number of test scores. The third line of space-separated integers describes scores.

Constraints

- $1 \leq \text{length of firstName, length of lastName} \leq 10$
- $\text{length of idNumber} \equiv 7$
- $0 \leq score \leq 100$

Output Format

Output is handled by the locked stub code. Your output will be correct if your *Student* class constructor and *calculate()* method are properly implemented.

Sample Input

```
Heraldo Memelli 8135627
2
100 80
```

Sample Output

```
Name: Memelli, Heraldo
ID: 8135627
Grade: O
```

Explanation

This student had 2 scores to average: 100 and 80. The student's average grade is (100+80)/2= 90. An average grade of corresponds to the letter grade O, so the *calculate()* method should return the character 'O'.

Code-

```
class Person:
    def __init__(self, firstName, lastName, idNumber):
        self.firstName = firstName
        self.lastName = lastName
        self.idNumber = idNumber
    def printPerson(self):
        print("Name:", self.lastName + ",", self.firstName)
        print("ID:", self.idNumber)

class Student(Person):
  def __init__(self, firstName, lastName, idNumber, scores):
        Person.__init__(self, firstName, lastName, idNumber)
        self.scores = []
        for s in scores:
            self.scores.append(int(s))
  def calculate(self):

        a = float(sum(self.scores))/len(self.scores)
        if a < 40:
            return 'T'
        elif a < 55:
            return 'D'
        elif a < 70:
            return 'P'
        elif a < 80:
            return 'A'
        elif a < 90:
            return 'E'
        else:
            return 'O'

line = input().split()
firstName = line[0]
lastName = line[1]
idNum = line[2]
numScores = int(input()) # not needed for Python
scores = list( map(int, input().split()) )
s = Student(firstName, lastName, idNum, scores)
s.printPerson()
print("Grade:", s.calculate())
```

(79) Power Of A Number

--Book by Charles Dierbach #16

Using the Binary approach, develop a solution to calculate the power of a number, given the number and the exponent.

Code-

```
def pow_e(p,q):
    r=q//2
    return (p**r)*(p**r)

def pow_o(p,q):
    r=int(q/2)
    return (p**r)*(p**(q-r))


p,q=int(input()),int(input())
if q%2==0:
    print(pow_e(p,q))
else:
    print(pow_o(p,q))
```

(80)What is your first name?

-HackerRank# 47

You are given the firstname and lastname of a person on two different lines.

Your task is to read them and print the following:

Hello firstname lastname! You just delved into python.

Input Format

The first line contains the first name, and the second line contains the last name.

Constraints

The length of the first and last name ≤ 10.

Output Format

Print the output as mentioned above.

Sample Input 0

Ross
Taylor

Sample Output 0

Hello Ross Taylor! You just delved into python.

Explanation 0

The input read by the program is stored as a string data type. A string is a

collection of characters.

Code –

```
def print_full_name(a, b):
    b=b+"!"
    print("Hello",a,b,"You just delved into python.")

if __name__ == '__main__':
    first_name = input()
    last_name = input()
    print_full_name(first_name, last_name)
```

(81) Functions

Given three points or length of three sides, write an algorithm and the subsequent Python code to check if they can form a triangle or not.

Three points can form a triangle if they do not fall in a straight line. Secondly, if the sum of the length of any two sides is less than the length of the third sides then they cannot form a triangle.

Use a function *coordinate_check(p1, p2, p3)* which returns true if all the three points do not lie on a straight line else returns false.

Call another function named *side_check(s1, s2, s3)* which returns true if the sum of the length of any two sides is greater than the length of the third sides else returns false.

If the user is given the option to choose either coordinate entries or side length entry for a triangle to check if a triangle is possible or not, then your code should return ‘Possible’ or ‘Not Possible’ based on the above-mentioned conditions.

Input format:

In the first line, enter 1 if coordinates are to be entered else 2 if side lengths are to be entered. Any other entry must print ‘Invalid Entry’ and should not proceed further

In the second line, space-separated x-coordinate and y-coordinate of point-1 or length of side-1

In the third line, space-separated x-coordinate and y-coordinate of point-2 or length of side-2

In the fourth line, space-separated x-coordinate and y-coordinate of point-3 or length of side-3

Output format:

Possible, Not Possible, or Invalid Entry

Sample Input 1:
1
2 4
-1 2
0 0
Sample Output 1:
Possible
Sample Input 2:
1
0 0
1 1
-1 -1
Sample Output 2:
Not Possible
Sample Input 3:
2
5
8
10
Sample Output 3:
Possible
Sample Input 4:
2
5
8
20
Sample Output 4:
Not Possible
Sample Input 5:
3
Sample Output 5:
Invalid Entry

Code-

```
def coordinate_check(a,b,c):
    a=[int(i) for i in a.split()]
    b=[int(i) for i in b.split()]
    c=[int(i) for i in c.split()]
    x=((a[0]-b[0])*(b[1]-c[1]))-((b[0]-c[0])*(a[1]-b[1]))#formula for area
    if x==0:
        return False
    return True

def side_check(a,b,c):
    if (a+b)>c:
        if (b+c)>a:
            if (c+a)>b:
                return True
    return False

n=int(input())
if n==1:
    a=input()
    b=input()
    c=input()
    if coordinate_check(a,b,c):
        print("Possible")
    else:
        print("Not Possible")
elif n==2:
    a=int(input())
    b=int(input())
    c=int(input())
    if side_check(a,b,c):
        print("Possible")
    else:
        print("Not Possible")
else:
    print("Invalid Entry")
```

The End