**­VPROPEL CODES :**

#### 1. Sum of digit powers

Sum of digit powers of a five-digit number is defined as follows:

1^d1 + 2^d2 + 3^d3 + 4^d4 + 5^d5

Where d1 is the right most digit and d5 is the leftmost digit.

For example, Sum of digit powers of 71694 is

1^4 + 2^9 + 3^6 + 4^1 + 5^7

= 79371

Given a five digit number n, write a code to find the sum of digit powers of n.

**Input Format**

First line contains the number, n

**Output Format**

Print the sum of digit powers of n

ANS.

**My code:**

n=int(input())

d1=n%10

d2=((n%100)-(n%10))/10

d3=((n%1000)-(n%100))/100

d4=((n%10000)-(n%1000))/1000

d5=((n)-(n%10000))/10000

c=(5\*\*d5)+(4\*\*d4)+(3\*\*d3)+(2\*\*d2)+(1\*\*d1)

print(round(c))

**Alternate code :**

n=int(input)

m=str(n)

a=1\*\*(int(m[4])

(similarly for all digits of the no.)

s=a+b+c+d+e

print(s)

#### 2. Weight of Salt

Sugar and salt are kept in two bowls A and B. Given the weight of r% of sugar in bowl A and ratio of weights (in gms) of sugar and salt in bowls A and B respectively, write a code to find the weight of salt (in gms) in bowl B.

For example, if weight of 30% of sugar is 100 gms and the ratio of sugar and salt in bowl A and B is 4:5 then the weight of salt is 416.67 (rounded to two decimal places).

**Input Format**

First line contains the value of r, the percentage of sugar for which weight is given

Next line contains the value of weight of r% of sugar

Next line contains the ratio part of sugar

Next line contains the ratio part of salt

**Output Format**

Print the quantity of salt in bowl B (Print only two decimal places)

**Note:**

To print only two decimal places of variable weight, use:

ANS.

r = int(input())

n = int(input())

partsugar=int(input())

partsalt=int(input())

sugarwt = 100\*(n/r)

wtsaltbowlb = (sugarwt)\*((partsalt)/(partsugar))

print(format(wtsaltbowlb,"0.2f"))

3.

Issue Train Tickets

Four persons are standing in a queue to book takal ticket for a particular train. At a particular point of time, the number of seats avaiable gets reduced to three. So the ticket issuing person decides to give tickets for three seniormost (age wise) persons among the four persons waiting to get the ticket. Each of them in queue had been issued a token. Given the token number issued and the age of the four people, write a code to print the token number of persons for whom tickets will be issued. Print the token numbers in the order same as the input order. For example, if the token numbers and age are as follows:

101, 45

104, 56

109, 38

125, 67

Then print token numbers 101, 104 and 125.

**Input Format**

First line contains the token number of person1

Next line contains the age of person1

Next line contains the token number of person2

Next line contains the age of person2

Next line contains the token number of person3

Next line contains the age of person3

Next line contains the token number of person4

Next line contains the age of person4

**Output Format**

Print token number of persons who will get tickets

**Note:**Print token numbers in the order same as input and print one token number in one line

ANS.

a=int(input())

a1=int(input())

b=int(input())

b1=int(input())

c=int(input())

c1=int(input())

d=int(input())

d1=int(input())

e=[a,b,c,d]

f={a1:a,b1:b,c1:c,d1:d}

g=[a1,b1,c1,d1]

i=f[min(g)]

e.remove(i)

for j in e:

print(j)

**EXPLANATION OF THE CODE IN 3.**

* IN THE PREVIOUS CODE, WE HAVE INPUTTED FIRST THE TOKEN NO. OF EACH INDIVIDUAL AND THEN HIS AGE.
* THEN WE HAVE CREATED A LIST THAT CONTAINS ALL THE TOKEN NOS.
* THEN WE HAVE MADE A DICTIONARY WHICH HAS KEYS AS AGES AND TOKENS AS THE MAIN THINGS OF THE AGES (ANALOGY- WORD IN A DICTIONARY-KEY AND ITS MEANING-MAIN THING).
* NOW, WE HAVE CREATED A SEPARATE LIST OF KEYS.
* AFTER THIS WE FIND THE MINIMUM AGE OUT OF THE KEYS AND USE THE DICTIONARY TO ACCESS ITS MAIN THINGS I.E TOKEN (SYNTAX IN THE CODE) AND STORE THIS MAIN THING WE FOUND IN A VARIABLE.
* NOW WE REMOVE THIS MAIN THING OR TOKEN FROM THE LIST OF TOKENS BY USING THE VARIABLE.
* THEN WE PRINT THE REMAINING LIST OF TOKENS IN THE EXACT SAME ORDER AS WE INPUTTED (ACTUALLY YOU SHOULD ENTER THE TOKENS IN A PARTICULAR ORDER AND LIST THOSE TOKENS IN THE LIST OF TOKENS IN THAT VERY SAME ORDER).

#### 4. Reverse right most digit of sum number

Given two three digit numbers n1 and n2, reverse right most digit of sum (RMDS) number is formed as follows:

(i) RMDS number is also a three digit number

(ii) RMDS number is formed by adding the digits of the two numbers n1 and n2 from right most digit to left most digit

(iii) First digit of RMDS number is formed by adding third digits of n1 and n2 and second digit of RMDS number is formed by adding second digits of n1 and n2 and third digit of RMDS number is formed by adding first digits of n1 and n2

(iv) Whenever sum of the digits of the numbers n1 and n2 is a two digit number, digit of RMDS number is the digit in the unit position of the sum.

(iv) When digit in unit position of sum of the third digits of n1 and n2 is '0', replace it with '9' (i.e.) first digit of RMDS number should not be zero so replace it by '9' if it happens to be '0' by the above process

For example, if n1 is 345 and n2 is 847 then RMDS number is 281 and when n1 is 345 and n2 is 875 the RMDS is 911.

**Input Format**

First line contains the number n1

Next line contains number n2

**Output Format**

Print the RMDS number formed from n1 and n2

ANS.

n1=int(input())

n2=int(input())

a1=n1%10

a2=(n1%100-a1)/10

a3=(n1-n1%100)/100

b1=n2%10

b2=(n2%100-b1)/10

b3=(n2-n2%100)/100

a=a1+b1

b=a2+b2

c=a3+b3

x=a%10

y=b%10

z=c%10

if (x==0):

x=9

if (z==0):

z=9

h=100\*x+10\*y+z

print(round(h))

#### **5.Bill amount to be paid**

As a Diwali offer, a textile shop gives a discount based on the total bill amount as follows:

|  |  |
| --- | --- |
| **Total Bill Amount** | **Discount** |
| >₹ 5000 | 25% |
| ₹3000 to ₹4999 | 15% |
| ₹1000 to ₹2999 | 10% |
| < ₹1000 | 5% |

Given the total amount of purchase by a customer, write a code to find the amount payable by the customer. Amount payable should only be integers so round up if decimal part of amount payable is >= 0.50 and round down otherwise. For example, if the total bill amount is ₹ 6437 then amount payable is ₹4828 and if total bill amount is ₹3591 then amount payable is ₹3052.

**Note:**

Smallest integer greater than 'n' can be found in Python as follows:

import math

m = math.ceil(n)

Largest integer lesser than 'n' can be found in Python as follows:

import math

m = math.floor(n)

**Hint:**

In Python, integer part of a real number 'n' can be got as int(n)

**Input Format**

First line contains the total bill amount of a customer in rupees (₹)

**Output Format**

Print the amount payable, only integers values can be paid so round up if decimal part is >= 0.5 and round down otherwise

ANS.

n=float(input())

t=0

if(n>5000):

t=n/4

elif(n>3000 and n<4999):

t=(3\*n)/20

elif(n>1000 and n<2999):

t=n/10

elif(n<1000):

t=n/20

k=n-t

i=int(k)

j=k-int(k)

if(j>=0.5):

import math

s=math.ceil(k)

elif(j<0.5):

import math

s=math.floor(k)

elif (j==0):

s=k

print(s)

NOTE: IF YOU WANT TO MENTION A RANGE WHICH HAS ITS LIMITS AS INTEGERS BUT THE NUMBERS COMPARED CAN BE FLOAT VALUES AS WELL DON’T USE “&” SYMBOL INSTEAD USE AND WORD.JUST REMEMBER THAT USE AND WORD TO MENTION RANGES IN IF ELSE WHEN TALKING ABOUT FLOAT VALUES.

6.Move Bits

All numerical and textual data are internally stored in binary form. Binary representation

of data contains only 0’s and 1’s. For example, number 35 is internally represented as

shown below, here only ten bits are shown remaining bits that to the left are 0’s.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

‘Bitwise and’ helps us to check if a particular bit of a number ‘n’ is 1. ‘Bitwise and’ of ‘n’

with the values ‘v’ returns a non-zero value if the bit at the given position ’p’ is 1

and returns zero otherwise; observe the pattern in this

|  |  |
| --- | --- |
| **Position (p)** | **Value (v)** |
| 1 | 1 |
| 2 | 2 |
| 3 | 4 |
| 4 | 8 |
| 5 | 16 |
| .... | .... |
| 10 | 1024 |
| 11 | 2048 |
| .... | .... |
| 20 | 1048576 |
| ... | .... |

We can move bits right or left. Moving 35 which is shown in above figure by two positions

left will have a bit representation as shown below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

Moving 35 by two positions right will have a bit representation as shown below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Given a number ‘n’ and a position ‘p’, write a code to check if the pth bit of n is 1.

If pth bit of ‘n’ is 1 then move left the bits of ‘n’ by ‘p-1’ positions and move right the

bits of ‘n’ by ‘p-1’ positions.

For example, when ‘n’ is 35 and ‘m’ is 2, the bit is 1 therefore do a right shift by one position

and the number will be 17.

When ‘n’ is 35 and ‘m’ is 3, the bit is 0 therefore do a left shift by two positions and the

number will be 140.

**Note:**Python has got bitwise operators to perform the operations

**Input Format**

First line contains the number, n

Next line contains the number, m

**Output Format**

Print the left bit moved or right bit moved number

ANS.

n=int(input())

m=int(input())

a=0

if n & (1<<(m-1)):

a=n>>(m-1)

else :

a=n<<(m-1)

print (a)

NOTE: LEARN FROM HACKEREARTH AND GEEKS FOR GEEKS ABOUT BIT MANIPULATION.

7. Check Transmitted Value

XOR operation (⊕) is a bitwise operation that works on a pair of bits as follows:

|  |  |  |
| --- | --- | --- |
| **B1** | **B2** | **B1 ⊕ B2** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

When two integers are XORed the operation takes place for each bit in the numbers. For example when two integers 50 and 67 are XORed it works as below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **50** | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| **67** | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| **50⊕67 = 113** | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

An interesting fact about XOR operation is that when the resultant number is XORed with one of the input number, we will get the other input number. For example, 113 ⊕ 67 = 50 and 113 ⊕ 50 = 67.

A number ‘n’ is transmitted over a network from one city to another and for security reasons the number ‘n’ is encrypted by performing XOR operation with another number ‘x’. The transmission channel is not perfect. Sometimes during transmission, bits are shifted either left or right by few positions. Given the number recieved, the value of ‘x’ and original number ‘n’, write a code to check if the number is transmitted correctly. If the number is correctly transmitted then print “Correct transmission” otherwise print whether the number is shifted left or right and by how many positions (integer value).

For example, if the number received is 139 and the value of x is 67 and n is 50 then print “Shifted left by 2 bits”. If the number received is 79 and the value of x is 67 and n is 50 then print “Shifted right by 2 bits”. If the number received is 113 and the value of x is 67 and n is 50 then print “Correct transmission”.

**Note:**

1. When a number ‘n’ is shifted left by ‘m’ bits, it gets multiplied by 2m

2. When a number ‘n’ is shifted right by ‘m’ bits, it gets divided (integer division) by 2m

3. Logarithm value of a number ‘n’ to the base ‘b’ can be got in Python as follows:

import math

math.log(n,b)

**Input Format**

First line contains the number received

Next line contains the value of ‘x’

Next line contains the orginal number ‘n’

**Output Format**

Print the appropriate message about transmission

ANS.

y=int(input())

x=int(input())

n=int(input())

z=y^x

a=0

b=0

c=0

if (z>n):

a=z/n

import math

b=math.log(a,2)

c=int(b)

print("Shifted left by",c,"bits")

elif (z<n):

a=z/n

import math

b=math.log(a,2)

c=(-1)\*int(b)

print("Shifted right by",c,"bits")

else:

print("Correct transmission")

AND MY SILLY MISTAKE’S SOLN.:

THAT FIRST MESSGAGE IS SENT THEN IT SHIFTS BY ERRORS THEN DECRYPTED BY ENCRYPTION KEY AT THE LOCATION.

AND I WAS THINKING THAT ALREADY XOR WAS DONE AND MESSAGE WAS SENT.

ALTERNATE METHOD : BITS SHIFT DOESN’T WORK BUT WHY I DON’T KNOW.

CODE: ONE ALTERNATIVE TO THIS WAS ALSO THAT INSTEAD OF <,> SIGNS USE != NOT EQUAL SIGN.

y=int(input())  
x=int(input())  
n=int(input())  
z=y^x  
count=0  
if (z>n):  
 while (n<z):  
 n=n<<1  
 count=count+1  
 print("Shifted left by",count,"bits")  
elif (z<n):  
 while (n>z):  
 n=n>>1  
 count=count+1  
 print("Shifted right by",count,"bits")  
else:  
 print("Correct transmission")

THIS QUESTION HAS SOME VALUABLE INFO FOR STUDY IN ITSELF LOOK FOR IT.

GARV TANDON’S CODE:

1. **import** math
2. receivedNumber, encryptionKey, sentNumber = int(input()), int(input()), int(input())
4. receivedDecrypted = receivedNumber ^ encryptionKey
6. **if**(sentNumber == receivedDecrypted):
7. **print**("Correct transmission")
8. **elif**(receivedDecrypted > sentNumber):
9. t = receivedDecrypted/sentNumber
10. ans = math.log(t, 2)
11. **print**("Shifted left by {} bits".format(int(ans)))
12. **elif**(receivedDecrypted < sentNumber):
13. t = sentNumber//receivedDecrypted
14. ans = math.log(t, 2)
15. **print**("Shifted right by {} bits".format(int(ans)))

Club Meetings

A college has a dance and a music club. The members of both the clubs meet on January 1, 2020. Dance club members decided to meet every mth day and music club members decided to meet every nth day from Januray 1st.

Given the value of ‘m’ and ‘n’, find the number of times both the club members meet together in the month of January (include the meeting held on January 1st also).

For example, if the value of m is 2 and n is 3 then both the clubs will meet on 7th, 13th, 19th, 25th , 31st of January and the number of times they meet together is 6. And when m is 14 and n is 15 number of times they meet together is 1.

**Input Format**

First line contains the value of m

Next line contains the value of n

**Output Format**

Print the number of times both the times the club members will meet together

ANS.

m=int(input())

n=int(input())

import math

a=math.gcd(m,n)

lcm=(m\*n)/a

b=(31//lcm)+1

print(round(b))

LOOK AT IT THIS WAY THAT SUPPOSE I DIVIDE 6/2=3

THIS MEANS I GET THREE MULTIPLES OF 2 BEFORE OR EQUAL TO 6

2\*1=2

2\*2=4

2\*3=6

I JUST DID FLOOR DIVISON TO GET INTEGER OUTCOME.

LOOP WAY-

def compute\_lcm(x, y):

if x > y:

greater = x

else:

greater = y

while(True):

if((greater % x == 0) and (greater % y == 0)):

lcm = greater

break

greater += 1

return lcm

m = int(input())

n= int(input())

l = compute\_lcm(m,n)

s= 1

count = 0

while(s<=31):

count+=1

s+=l

print(count)

Split into Three Addend

Given three numbers n, a and b, write a code into split ‘n’ into three addends n1, n2 and n3 abiding the following conditions:

(i) Both a and b occurs atleast once as a addend

(ii) n1, n2, n3 ∈ {a, b} and n1>=n2>=n3

(iii) n = n1+n2+n3

If ‘n’ cannot be split into three addends as per conditions given above, print “Cannot be written”.

For example,

(i) if n = 130, a= 50, b = 30 then n1 = 50, n2 = 50 and n3 = 30

(ii) if n = 110, a = 50 and b = 30 then n1 = 50, n2 = 30 and n3 = 30

(iii) if n = 110, a = 20 and b = 30 then print Cannot be written

**Input Format**

First line contains the value of n

Next line contains the value of a

Next line contains the value of b

**Output Format**

Print the value of n1, n2 and n3 in a line separated by a space. No space at the end of the line

ANS.

n=int(input())

a=int(input())

b=int(input())

c=n-(a+b)

n1=0

n2=0

n3=0

if (c==a and a>b):

n1=a

n2=a

n3=b

print(n1,n2,n3,end=" ")

elif (c==a and a<b):

n1=b

n2=a

n3=a

print(n1,n2,n3,end=" ")

elif (c==b and a>b):

n1=a

n2=b

n3=b

print(n1,n2,n3,end=" ")

elif (c==b and a<b):

n1=b

n2=b

n3=a

print(n1,n2,n3,end=" ")

else :

print("Cannot be written")

Q. INPUT TWO NUMBERS WITH SAME NO. OF DIGITS AND THEN TELL AT WHICH PLACES THE DIGITS ARE SAME.

ANS. My Code-

1=int(input())  
n2=int(input())  
m1=str(n1)  
m2=str(n2)  
i=0  
count=0  
boom=0  
if(m1[len(m1)-1]==m2[len(m2)-1]):  
 print("Same at 1's position")  
 boom=1  
for i in range (len(m1)-2,-1,-1):  
 if(m1[i]==m2[i]):  
 print("Same at %dth postion" %(10\*\*(len(m1)-1-i)))  
 count+=1  
if(count==0 & boom==0):  
 print("No digits are same")

Identifying Quadrilateral

Following are the properties of a quadrilateral and few types of quadrilaterals:

1. Sum of four angles of any quadrilateral is 360°

2. All the angles of a rectangle are 90°

3. Opposite angles are equal in a parallelogram

4. For a trapezium, two angles on the same side are supplementary

Given four numbers, A, B, C and D write a code to check if they can be angles of a quadrilateral with the angles A opposite to C and B opposite to D.

(i) If the angle property of quadrilateral is not satisfied then print “No Quadrilateral”

(ii) If the angles satisfy the properties of special type of quadrilateral then print either Rectangle, Parallelogram or Trapezium appropriately.

(iii) If the angles are not satisfying the properties of Rectangle, Parallelogram or Trapezium then print “Quadrilateral”.

For example, if the value of A, B, C and D is 60, 120, 60, 120 then print “Parallelogram” and if values of A, B, C and D is 70, 100, 110, 80 then print “Quadrilateral”

**Input Format**

First line contains value of A

Next line contains value of B

Next line contains value of C

Next line contains value of D

**Output Format**

Print either No Quadrilateral, Rectangle, Parallelogram, Trapezium or Quadrilateral

ANS.

a=int(input())

b=int(input())

c=int(input())

d=int(input())

e=a+b+c+d

f=a+b

g=c+d

if (e!=360):

print("No Quadrilateral")

elif (a==90 and b==90 and c==90 and d==90):

print("Rectangle")

elif (a==c and b==d and a!=b and c!=d):

print("Parallelogram")

elif (f==180 and g==180 and a!=b and a!=c and a!=d and b!=c and b!=d and c!=d):

print("Trapezium")

else :

print("Quadrilateral")

NOTE:

USE ‘AND’ IN CASE OF CONDITIONS WHERE YOU REQUIRE TO USE MANY CONDITIONS NOT & IN IF.

Numbers with Common Greatest Factor

Greatest factor of a number n, is the largest factor of ‘n’ excluding itself. For example, greatest factor of 9 is 3, 15 is 5 and so on.

Given two numbers, n1 and n2, write a code to check if the numbers have common greatest factor. If they have common greatest factor then print the factor and print ‘No’ otherwise.

For example, 6 and 9 have common greatest factor, 3, so print 3. If input is 15 and 21 then print 'No' as greatest factor of 15 is 5 and greatest factor of 21 is 7.

**Input Format**

First line contains the number, n1

Next line contains the number, n2

**Output Format**

Print the common greatest factor if it exist and print No otherwise

ANS.

n1=int(input())

n2=int(input())

a=0

b=0

c=0

d=0

for i in range(1,(n1//2)+1,1):

a=n1%i

if(a==0):

b=i

for j in range(1,(n2//2)+1,1):

c=n2%j

if(c==0):

d=j

if(b==d):

print(int(b))

else:

print("No")

Number of Weeks to Get a Video Game

Varun likes to get a video game of cost Rs. x, but his father doesn’t want to get him immediately. So he tells varun that he will give a pockey money of Rs. p every week which he can save and get the game. But varun wanted to get it faster so he pleaded his father to increase the pocket money evey week by Rs. q. His father agrees. Given the cost of the video game, x, pocket money initially decided p, increase in pocket money every week q, write a code to find the number of weeks that varun will take to get the video game.

For example, if the cost of the video game is Rs. 100, initial pocket money agreed is Rs. 20 and increase in pocket money agreed for every week is Rs. 2 then varun will take 5 weeks to get the video game.

**Input Format**

First line contains the cost of the game, x

Next line contains the pocket money given per week, p

Next line contains the increase in pocket money every week, q

**Output Format**

Print the number of weeks that varun will take to buy the video game

ANS.

x=float(input())

p=float(input())

q=float(input())

total=0

count=0

while(x>total):

p+=q

total+=p

count+=1

print(count+1)

Sum of Three’s and Five’s

A number greater than eight can be written as sum of 3’s and 5’s. Given a number, n, write a code to check if n can be written as sum of only 3’s, sum of only 5’s or sum of ‘p’ number of 3’s and ‘q’ number of 5’s where p is always maximum possible value.

For example, 9 can be written as sum of only 3’s, 25 can be written as sum

of only 5’s.

If n is less than 8 then print ‘Invalid input’, if a number can be written as sum of only 3’s and only 5’s then consider sum of only 3’s. Use number names of 3 and 5 while printing.

**Examples:**

(i) **Input:** n = 15

**Output:** Only three’s

(ii) **Input:**n = 127

**Output:** 39 three's and 2 five's

**Input Format**

First line contains the number, n

**Output Format**

If number n can be written using only 3’s then print Only three’s, n can be written using only 5’s then print Only five’s otherwise print the number of three’s and number of five’s

Use number names of 3 and 5 while printing.

ANS.

n=int(input())

if (n<8):

print("Invalid input")

elif (n%3==0):

print("Only three's")

elif (n%5==0):

print("Only five's")

elif (n%3==0 and n%5==0):

print("Only three's")

else :

for i in range (0,n):

a=n-i

if (a%3==0 and i%5==0):

break;

b=a//3

c=i//5

print("%d three's and %d five's" %(b,c))

Squeezable to Four Numbers

Squeezing is a process in which the digits of a number is squared and summed up. For example, if the number is 345 then after squeezing, we get 32+42+52 = 50.

Repeated squeezing is a process in which squeezing is repeated for a number ‘n’, for some number of iteration. For example, when repeated squeezing is done for two iterations for 345. It looks as follows:

345 -> 32+42+52 = 50

50 -> 52 + 02 = 25

A number n, is said to squeezable to four, if it is possible to get 4 after a squeezing operation in number of iterations less than or equal to 10. For example, 24 can be sequeezed to 4 in two iterations

24 -> 22 + 42 -> 20

20 -> 22 + 02->4

And 500 can be sequeezed to 4 in eight iteration as shown below:

500-> 52 + 02 + 02 -> 25

25 -> 22 + 52 -> 29

29 -> 22 + 92 -> 85

85 -> 82+ 52->89

89 -> 82 + 92 -> 145

145 -> 12 + 42 + 52 -> 42

42 -> 42 + 22 -> 20

20 -> 22 + 02 -> 4

Given a number, n, write a code to print the number of iterations required if 'n' is squeezable to four (in number of iterations <=10) and print 'No' otherwise

**Boundary Conditions**

4 < n <= 1000000000

**Input Format**

First line contains the number, n

**Output Format**

Print the number of iterations required if the number is squeezable to four and print No otherwise

ANS.

def totaler (a):

total=0

while (a>0):

rem=a%10

sq=rem\*rem

total = total + sq

a = a//10

return(total)

n=int(input())

count=0

d=0

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

d=totaler(n)

n=d

count+=1

if (d==4):

break;

if (count<11 and d==4 ):

print(count)

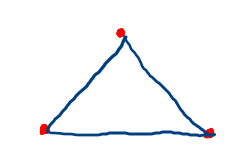
else:

print("No")

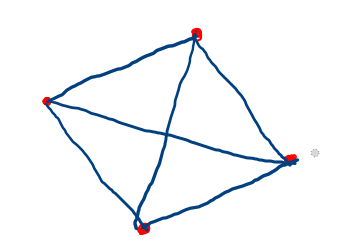
Lines Connecting all Points

**There was a mistake in today's test case that it would have not worked for all the possible ways of writing the solution. Now it is corrected. Apologies for that**

Total number of lines that can be drawn by connecting ‘n’ number of points is nc2. For example given three points, three lines can be drawn.



With four points, six lines can be drawn as shown below



Given the total number of lines that can be drawn, n, write a code to find the number of points which can be connected by ‘n’ lines. For example, if the number of lines drawn is 28 then the number of points is 8. If number of lines in 210, the number of points is 21.

**Boundary Conditions**

0 <n <1000000

**Input Format**

First line contains the number of lines that can be drawn, n

**Output Format**

Print the number of points that are connected by ‘n’ lines

 ANS.

NOTE: JUST REVERSE THE FORMULA GIVEN AND SOLVE THE QUADRATIC EQUATION.

b=int(input())

import cmath

n1=(1+cmath.sqrt(1+8\*b))/2

n2=(1-cmath.sqrt(1+8\*b))/2

if (int(n1.real)>0) :

print(int(n1.real))

elif (int(n2.real)>0) :

print(int(n2.real))

Chickens in Enclosures

In a small poultry farm there are three enclosures. There are odd number of chickens in the farm and each enclosure can contain odd number of chickens only. Second enclosure is larger than the first enclosure and third enclosure is larger than the second enclosure. Given the total number of chickens ‘n’ in the poultry farm and the minimum number of chickens that can be put in the first enclosure, ‘m’, write a code to print all possible ways in which the chickens can be kept.

For example, if the total number of chicken in the farm is 27 and the minimum number of chickens that can be kept in first enclosure is 3 there are seven ways to keep the chickens as shown below:

3 5 19

3 7 17

3 9 15

3 11 13

5 7 15

5 9 13

7 9 11

If the ‘n’ chickens cannot be kept with minimum number of ‘m’ chickens in the first enclosure, then print ‘No way’. For example, if number of chicken in the poultry farm is 11 and minimum number of chickens to be kept in first enclosure is 3 then print ‘No way’

**Input Format**

First line contains the number of chicken the polutry farm, n

Next line contains the minimum number of chickens that can be placed in first enclosure, m

**Output Format**

Print the possible ways of placing the chickens.

Print each way of placing the chicken in a line, values of each enclosure separated by a space

Printing order is in ascending order of enclosure 1 and then ascending order of enclosure 2

ANS.

n=int(input())

m=int(input())

o=m

p=m+2

a=[]

while (o<p):

for p in range (m+2,n+1,2):

for q in range (p+2,n+1,2):

if (o<p and o+p+q==n):

print(o,p,q)

a.append([o,p,q])

o+=2

if (len(a)==0):

print("No way")

EXPLANATION- (REQUIRED)

A multiplication is said to be unique when the digits of the multiplicand, multiplier and product are different and digits from 1 to 9 are present in it.

For example, 4\*1963 = 7852 is unique multiplication as all the nine digits from 1 to 9 are present in it and the digits in the multiplicand, multiplier and product are different.

Given the multiplicand and the multiplier, write a code to check if it is a unique multiplication. Print Yes for unique multiplication and No otherwise

**Input Format**

First line contains the value of multiplicand, n

First line contains the value of multiplier, m

**Output Format**

Print Yes for unique multiplication and No otherwise

ANS.

n=int(input())

m=int(input())

p=n\*m

n1=str(n)

n2=str(m)

n3=str(p)

l1=[]

l2=[]

l3=[]

l1[:0]=n1

l2[:0]=n2

l3[:0]=n3

l4=l1+l2+l3

l5=sorted(l4)

l6=['1','2','3','4','5','6','7','8','9']

if l5==l6:

print("Yes")

else:

print("No")

#### **Complete n-Digit Factor**

A number is said to be a complete ‘n’ digit factor, if it is a factor of all ‘n’ digit numbers when concatenated (joined to right) to itself.

For example, 7 is a complete 3-digit factor as it divides all three digit numbers from 100 to 999 when concatenated to itself (i.e. 100100, 101101,102102, ... ..., 998998, 999999).

Given the value of n and m, write a code to generate all numbers from 2 to m (both inclusive) that are complete n-digit factor and print ‘No complete factors’ otherwise.

For example, if n is 3 and m is 15 then print 7, 11, 13 and if n is 3 and m is then print ‘No complete factors’

**Boundary Conditions:** 2<n<9

**Note:**The code needs to optimized to complete execution within the time bound given for the problem

**Input Format**

First line contain the number of digits, n

Next line contains the maximum value of m that has to be checked for Complete n-Digit Factor

**Output Format**

Print each Complete n-Digit Factor in one line and print No complete factors otherwise

ANS.

#### **Vowel Based Blended Words**

In English, blend words are words which are formed by combining two English words. Vowel based blended words are one which are formed by joining the part of the first word before a vowel in it and part of the second word starting from the same vowel as in first word.

For example, spork is a vowel based blending word formed from spoon and fork as shown below:

spoon + fork – spork

When more than one vowel is present in the words, consider the first occurrence of the vowels.

Given two words, write a code to form the vowel based blended word from them

**Input Format**

First line contains the first word, w1

Next line contains the second word, w2

**Output Format**

Print the vowel based blended words formed

ANS.

w1=str(input())

w2=str(input())

for i in range (0,len(w1)-1):

if (w1[i]=='a' or w1[i]=='e' or w1[i]=='i' or w1[i]=='o' or w1[i]=='u'):

temp=i

break;

for j in range (0,len(w2)-1):

if (w2[j]=='a' or w2[j]=='e' or w2[j]=='i' or w2[j]=='o' or w2[j]=='u'):

temp1=j

break;

w3=w1[:i]

w4=w2[j:len(w2):1]

w5=w3+w4

print(w5)

STRING SLICING FUNDA- stringname[start:end:step]

#### **Same Letter Distances**

Same letter distance is the number of characters between two consecutive occurrances of the same character. Given a string, write a code to find the letter that has most frequently occurred in it. And also find same letter distance between the occurrances of the most frequently occurred character. If distance is zero then print ‘No characters’, if distance is 1 then print ‘1 character’ and if distance is ‘d’, greater than 1 then print ‘d characters’.

For example, if the string is ‘accomodation’, the most frequently occurred character is ‘o’ and the same letter distances are

1 character

4 characters

If the string is ‘bookkeeper’, the most frequently occurred character is ‘e’ and the same letter distances are

No characters

1 character

If more than one character occurs maximum number of times then consider the first character in the string.

**Input Format**

First line contains the string, w

**Output Format**

Print the letter that has occurred most frequently, ch

Print the same letter distance for consecutive occurrences of ch in next few lines

ANS. PART FOR CALCULATING THE MOST FREQUENT LETTER-

w=str(input())

freq = {}

for i in w:

if i in freq:

freq[i] += 1

else:

freq[i] = 1

res = max(freq, key = freq.get)

ch=str(res)

print(ch)

OTHER PART :

NOT DONE BUT IN PROCESS

for j in range (0,(len(w)//2)):

for k in range (len(w)-1,(len(w)//2)+1,-1):

if (w[j]==w[k]):

if((j-2)==1):

print("1 character")

elif((j-2)!=1):

print("%d characters" %(j-1))

NEED TO FIND THE RIGHT LOGIC FOR THIS PART.

FAILED TEST CASES :

Input

yellowwood

Expected output

o

2 characters

No characters

Your Program Output

o

3 characters

3 characters

THIS GETS PASSED WHEN j-2 -> j-1

Input

accomodation

Expected output

o

1 character

4 characters

Your Program Output

o

1 character

3 characters

**ANS.**

**from collections import Counter**

**w = input()**

**x = Counter(w)**

**max,max\_c = 0,''**

**for i in x.keys():**

**if x[i]>max: max,max\_c = x[i],i**

**pos = []**

**for i in range(len(w)):**

**if w[i] == max\_c: pos.append(i)**

**print(max\_c)**

**for i in range(len(pos)-1):**

**a = pos[i+1]-pos[i]-1**

**if a == 0: print("No characters")**

**elif a == 1: print("1 character")**

**else:**

**print(a,"characters",sep=' ')**

Chintu’s internship project

Chintu has finally got his dream internship in one of the world’s biggest tech companies, Hooli. Chintu was assigned a project for his internship. A part of his project involved extracting tag information from a paragraph. Tag information can then be looked into a database to find what it refers to. The format for a tag is:  
word[TAG]  
Here, the word is made up of English alphabets and can include both lowercase and uppercase letters(meaning a-z and A-Z only).  
TAG is a string contained inside square brackets which contains an ID in uppercase and a number concatenated by ‘-’.  
Example Tags: [TEST-123] , [ABC-123], [INV-01]  
For example:  
Consider we have a paragraph:  
gafkjen[ABC-123] this is some test data,[ABC-2131]klsd final Lorem ipsum dolor sit amet, consectetur adipiscing[TEST-1234],elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate[ID-2341] velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

In this example,  
The tagged words are:  
gafkjen[ABC-123]  
adipiscing[TEST-1234]  
voluptate[ID-2341]  
If you are wondering why [ABC-2131] is not included is because it is not following a word, its following a comma, so we consider it as a part of the paragraph itself.  
**Note:** ​ If no valid tagged information is found, print ​ -1

**Hint:** **Using re package will lead to a shorter solution**

**Input Format:**  
Input is a string of length not more than 10​5​ characters.  
**Output Format:**  
For every word that is tagged, print a line containing the word and the tag separated by space. The tag should not contain square brackets. The order of tagged words must be preserved,  
meaning that the tagged word which comes first should be printed first.  
**Sample Input 1:**  
gafkjen[ABC-123] this is some test data,[ABC-2131]klsd final Lorem ipsum dolor sit amet, consectetur adipiscing[TEST-1234],elit, sed do eiusmod tempor incididunt ut labore et doloremagna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate[ID-2341] velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.  
**Sample output 1:**  
gafkjen ABC-123  
adipiscing TEST-1234  
voluptate ID-2341  
**Explanation 1:**​ Given in the problem statement itself.  
**Sample Input 2:**  
[ABC-123][TEST-123][TEST-456]  
**Sample Output 2:**  
-1  
**Explanation:**  
The string does not match any tagged words because nothing has the format word[TAG], so we print -1.

ANS. YET TO BE FOUND

#### **Splicing of Strings**

Let S1 and S2 be any two strings with few symbols in common. If there are duplicates in S1 or S2 then remove them. During the duplicate removal process retain the first occurence of the characters. For example, if element is the string then after duplicates removal it will be elmnt.

Crossover is the process of joining two strings S1 and S2 with pivot as one of the common characters of S1 and S2. If ‘α’ be a common character of S1 and S2 such that S1 = Prefix(S1)αSuffix(S1) and S2 = Prefix(S2)αSuffix(S2). Here Prefix(S1) is the sequence of characters that appear before α in S1 and Suffix(S1) is the sequence of characters that appear after α in S1. Then crossover(S1,S2, α) = Prefix(S1)+α+Suffix(S2) and Prefix(S2)+α+Suffix(S1). Splicing is the process of forming all posible new strings from S1 and S2 by performing crossover of S1 and S2 with all common characters.

For example, consider two strings abcdce and ecfgdc then after duplicate removal the strings are abcde and ecfgd. There are three common characters ‘c’ and ‘d’.

crossover(s1,s2,c) = abcfgd and ecde

crossover(s1,s2,d) = abcd and ecfgde

crossover(s1,s2,e) = abcdecfgd and e

and splicing (s1, s2) = abcfgd, ecde, abcd, ecfgde, abcdecfgd, e

**Input Format**

First line contains the first string, S1

Second line contains the second string, S1

**Output Format**

Print each newly formed string formed by splicing in sorted order. Print one new string in one line

ANS.

s1=str(input())

s2=str(input())

count1=0

count2=0

for i in range (0,len(s1)):

for j in range (i+1,len(s1)-1):

if (s1[i]==s1[j]):

count1+=1

s1changed=s1.replace(s1[j],"",count1-1)

for i1 in range(0,len(s2)):

for j1 in range (i1+1,len(s2)-1):

if (s2[i1]==s2[j1]):

count2+=1

s2changed=s2.replace(s2[j1],"",count2-1)

PROGRAM TO PRINT STRING WITHOUT DUPLICATE LETTER PART

BUT SECOND PART NOT FOUND AND REUSABILITY OF VARIABLES ISSUE

#### **Moving Averages**

Consider n numbers. Average of the given n numbers is the sum of all the numbers divided by n. Moving average of size k, k<=n, written as Mvk , is the average of the first k numbers among the n numbers. For a number with n digits, there will be n moving averages. Grand Moving average is the average of all the possible moving averages. If we have the numbers 1,7,8,9; moving average of size 1 is 1. Moving average of size 2 is (1+7)/2= 4. Moving average of size 3 is (1+7+8)/3=5.33; Moving average of size 4 is (1+7+8+9)/4 = 6.25. Grand moving average is (1+4+5.33+6.25)/4= 16.58/4=4.14

Given n numbers, Write the pseudocode and the code to compute the Grand Moving Average of the given numbers. The grand moving average should be printed in two decimal format.

**Input format**

First line contains the number of numbers, n

Next ‘n’ lines contain the numbers in the collection

**Output format:**

Print grand Moving Averages in two decimal format

**Illustration :**

**Input**

4

1

7

8

9

**Output**

4.14

ANS.

n=int(input())

l=[]

for i in range(0,n):

elm=int(input())

l.append(elm)

l1=[]

for i in range (0,len(l)):

l1.append(l[0:i+1])

l2=[]

import statistics

for i in range (0,len(l1)):

l2.append(statistics.mean(l1[i]))

avg=statistics.mean(l2)

print("{:.2f}".format(avg))

#### **Time Taken by Ant to Gather Food**

An ant has its nest in a tree which is at the bottom of a mountain. The mountain is irregular in structure that to travel from one point in the mountain to the other point in the mountain one has to climb up, move straight and climb down. Everyday this ant get down the tree, searches for food in the mountain and brings back the food to the nest. Ant moves with different speeds while it climbs down the tree, climbs up the tree, climbs down the mountain, climbs up the mountain and move straight on the mountain surface. Let P be the point in the mountain where food for the ant is available on a particular day.

Given the height of the tree, the distances to climb up, climb down and move straight to reach the point P from the nest of the ant and the different speeds with which the ant can move, determine the time taken by the ant to gather food from point P and save it in its nest. All the distances given in the problem are in meters and the speed of ant is given in meters/min. After calculation, ceil the time taken to next minute. Ceil function gives the next integer for all decimal places that is ceil(3.1) = 4 and ceil(3.9) = 4.

For example, if the height of the tree is 230 meters, to reach point P from the bottom tree the ant has to climb up 170 meters, move straight by 310 meters, climb down 217 meters and the ant can climb down the tree with speed 8 m/min, climb up the mountain with speed 6 m/min, move straight on mountain surface with speed 11m/min, climb down the mountain with speed 12 m/min and the ant can climb up the tree with speed 7m/min then the time taken by the ant to bring food to nest will be 3 hours and 35 minutes.

In python, ceil function for a variable ‘x’ may be used as

import math

math.ceil(x)

**Input Format**

First line contains the height of the tree, h

Next line contains the distance to climb up to reach point P

Next line contains the distance to move straight to reach point P

Next line contains the distance to climb down to reach point P

Next line contains the speed of the ant by which it can climb up the mountain

Next line contains the speed of the ant by which it can climb down the mountain

Next line contains the speed of the ant by which it can move straight on the mountain surface

Next line contains the speed of the ant by which it can climb up the tree

Next line contains the speed of the ant by which it can climb down the tree

**Output Format**

Print the number of hours and minutes (ceil to the next minute and separated by a space) taken by the ant to gather food from point P and bring back to the nest

ANS.

htree=float(input())

up=float(input())

straight=float(input())

down=float(input())

upspeed=float(input())

downspeed=float(input())

straightspeed=float(input())

upspeedtree=float(input())

downspeedtree=float(input())

totaltime=(htree/upspeedtree)+(htree/downspeedtree)+((up/upspeed)+(down/downspeed)+2\*(straight/straightspeed))+(down/upspeed)+(up/downspeed)

time=totaltime/60

timehrs=totaltime//60

timemins=(time-timehrs)\*60

import math

b=round(timehrs)

a=math.ceil(timemins)

print(b,a)

**C++**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int main(){**

**double h,d\_up,d\_str,d\_dwn,sp\_up,sp\_dwn,sp\_str,sp\_up\_tree,sp\_dwn\_tree;**

**cin>>h>>d\_up>>d\_str>>d\_dwn>>sp\_up>>sp\_dwn>>sp\_str>>sp\_up\_tree>>sp\_dwn\_tree;**

**double total\_time = h/sp\_up\_tree+h/sp\_dwn\_tree+2\*d\_str/sp\_str+d\_up/sp\_up+d\_dwn/sp\_dwn+d\_dwn/sp\_up+d\_up/sp\_dwn;**

**cout<<(int)(total\_time/60)<<" "<<ceil((total\_time/60-(int)total\_time/60)\*60);**

**}**

#### **Matrix Difference**

Matrix Difference is a binary operation which operates on two matrices of any dimensions and represented by the symbol ‘~’. It is defined as follows:

If M3 = M1~M2 then M3[i,j] = M1[i,j] if M1[i,j] ∉ M2 and M3[i,j] = 0 otherwise where M1[i,j] ∉ M2 indicates that M1[i,j] is not an element of matrix M2. Given two matrices M1 and M2 write a code to find M1~M2.

For example, if M1 is a 3X3 matrix as shown below:

4 5 7

1 9 11

12 3 2

and M2 is a 2X2 matrix as shown below:

15 5

11 2

then M3 should be

4 0 7

1 9 0

12 3 0

This operation is not commutative. M2~M1 will have the following elements:

15 0

0 0

**Note:**

1. Only valid input is given for the problem

2. **In t**he output, last element of each row is followed by a space

**Input Format**

First line contains the number of rows in M1, r1

Second line contains the number of columns in M1, c1

Next r1 lines contain the elements in one row of M1 separated by a space

Next line contains the number of rows in M2, r2

Next line contains the number of columns in M2, c2

Next r2 lines contain the elements in one row of M2 separated by a space

Note that only valid input is given for the problem

**Output Format**

Print each row of Matrix M3 in one line and elements in a row should be separated by a space

ANS.

r1=int(input())

c1=int(input())

m1,m2=[],[]

for i in range(r1):

m1.extend(input().split())

r2=int(input())

c2=int(input())

for i in range(r2):

m2.extend(input().split())

m3=[]

for i in m1:

if i in m2:

m3.append(0)

else:

m3.append(i)

a=0

for i in m3:

a+=1

print(i,end=" ")

if a%c1==0:

print()

String generation using W-function

Using an N-function, every letter of English alphabet (either upper case or lower case) can be represented as a number that corresponds to its position in the English alphabet. For example, ‘A’ is represented by 1, ‘z’ by 26. This is written as N(A)=1, N(a)=1. In the same way, using a W-function, a number k in the range, from 1 to 26 (both included) , is represented by the symbol whose positional index in the English alphabet, is k. This is written as W(25)=y.

From an integer n, two strings W1 and W2 may be generated in such a way that every character ‘c’ in W1 or W2, taken in order, is equal to W(k)=c where k is a single digit for generating W1 and it is a two digit number for generating W2, that occurs in order in the given integer ‘n’. When the two digit number, k is greater than 26 take the letter corresponding to modulus 26 of k.

Given an integer ‘n’ which does not have any zero as its digits, write an algorithm and the pseudocode to generate the two strings that can be formed from the number n using W-function.

Print both the words in lower case letters

For example, if n = 1234 then words are generated from n as follows.

1-2-3- 4 generates abcd

12-23-34 generates lwh

When n= 491

4-9-1 generates dia

49-91 – generates wm

**Input format :**

First line contains a number, n

**Output format**

Print W1 in first line

Print W2 in second line

ANS. MY CODE:

n=int(input())

m=str(n)

l=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

for i in m:

print(l[int(i)-1],end="")

k=[]

for i in range (0,len(m)-1):

a=m[i]+m[i+1]

k.append(int(a)%26)

print()

for i in k:

print(l[i-1],end="")

SHREYASH CODE :

n=input()

m=n+" "

nlist=[(int(i)+96) for i in n] #[97, 98, 99, 100]

nlist2=[(int((m[i]+m[i+1]).strip())+96) for i in range(len(n)-1)] #[108, 119, 130]

r1=[chr(i) for i in nlist] #['a', 'b', 'c', 'd']

r2=[chr(((i-96)%26)+96) for i in nlist2] #['l', 'w', 'h']

s1="".join(r1) #abcd

s2="".join(r2) #lwh

print(s1)

print(s2)

#### **Beautiful Numbers**

A sequence of numbers is called cute if all the numbers in the sequence are made of only two digits, 8 and 9.  
Example of a cute sequence is: 8, 9, 88, 89, 98, 99…. and so on.  
A number is called beautiful if it is divisible by any number which is part of the cute sequence.  
For example: 8 (divisible by 8), 9(divisible by 9), 889 (divisible by 889),  10668 (divisible by 889) are beautiful numbers. Given a number, n, write a code to print “beautiful” (without quotes) if it is divisible by any number that contains only 8 or 9 or both and print -1 otherwise.

**Input Format:**

Input contains one single line denoting the number, n

**Output Format:**  
Print "beautiful" if the number is a beautiful number and -1 otherwise

**Constraints:**  
1 <= N <= 999999999999999999

ANS.

n=int(input())

fib=[8,9]

s=0

for i in fib:

fib.append(i\*10+8)

fib.append(i\*10+9)

if len(str(i))==len(str(n)):

break;

n=int(n)

for i in fib:

if n%i==0:

print("beautiful")

s+=1

if s==0:

print(-1)

EXPLANATION:

ISS LOGIC MEIN SAARE CUTE NOS. NIKAL KE USS NO. KE HALF TAK DIVIDE KARAKE CHECK KARTE BAS

AGAR KISI SE BI DIVIDE HO JAATA TOH MAST OUTPUT

#### **Search Fibanocci Number**

Given an integer n, search for the set of all fibanocci numbers F = {f1∪f2∪f3∪...fn} where fi is the set of fibanocci numbers of length ‘i’ and present in the number ‘n’. A number in fi is formed by taking ‘i’ consecutive digits from ‘n’. For example, if the value of n is 121393 then f1= {1, 2, 3}, f2= {13, 21}, f3 = {}, f4 = {}, f5={} and f6={121393} hence F = {1, 2, 3, 13, 21, 121393}. If no fibanocci number is present then print None.

**Input Format**

First line contains a number, n

**Output Format**

Print all the fibanocci numbers present in n, in ascending order. Print one number in one line. If no fibanocci number is present then print None.

ANS.

n=int(input())

fib=[2,3]

flag=0

if '0' in str(n):

print(0)

flag=1

if '1' in str(n):

print(1)

flag=1

i=1

for j in fib:

x=fib[i-1]+fib[i]

fib.append(x)

if str(j) in str(n):

print(j)

flag=1

if fib[i]>n:

break;

i+=1

if flag==0:

print('None')

#### **Programmer Rank**

A university conducts a number of programming contests in their programming portal every day. The problems given in the portal are categorized into easy, medium and hard. A student scores 10 points when he solves an easy problem, he gets 30 points when solves a medium level problem and he scores 50 points when he solves a hard problem. The university decides to give a ranking for the students based on their active performance in the contests. Students with the highest score get the first rank, the second highest score gets the second rank and so on. If more than one student, say ‘k’ students get the same score then all get the same rank ‘r’ and the next student or students get ‘r+k’ th rank. For example, when there are six students with the following details:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Number of easy problems solved** | **Number of medium problems solved** | **Number of hard problems solved** |
| Student1 | 12 | 8 | 9 |
| Student2 | 11 | 8 | 11 |
| Student3 | 10 | 7 | 10 |
| Student4 | 13 | 7 | 8 |
| Student5 | 12 | 8 | 9 |
| Student6 | 21 | 6 | 7 |

Total points scored by the students are:

Student1 -  810

Student2  -  900

Student3 - 810

Student4 - 740

Student5 - 810

Student6 - 740

Rank given for the students are as follows:

Student1 - 2

Student2  - 1

Student3 - 2

Student4 - 5

Student5 - 2

Student6 - 5

Given the name of the student, the number of easy problems solved, the number of medium problems solved and the number of hard problems solved write a code to display the rank and the name of the students.

**Input Format**

The first line contains the number of students, n

Next ‘n’ lines contain the name, the number of easy problems solved, the number of medium problems solved, the number of hard problems solved separated by a space each.

**Output Format:**

Print rank of student and the name of the student separated by a space arranged in ascending order of rank. When the rank is same for more than one student print in ascending order of name. Print name and rank of one student in one line.

**Illustration**

For the example described above, input and output should be as follows:

**Input**

6

Student1 12 8 9

Student2 11 8 11

Student3 10 7 10

Student4 13 7 8

Student5 12 8 9

Student6 21 6 7

**Output**

1  Student2

2  Student1

2  Student3

2  Student5

5  Student4

5  Student6

ANS.

n=int(input())

rank={}

for i in range(n):

a=input().split()

marks=int(a[1])\*10+int(a[2])\*30+int(a[3])\*50

try:

rank[marks].append(a[0])

except KeyError:

rank[marks]=[a[0]]

keys=list(rank.keys())

keys.sort(reverse=True)

f,k=1,1

for i in keys:

for j in rank[i]:

print(f," ",j)

k+=1

f=k

#### **Game of Dragons**

IronMan is stuck on a level of the Infinity War game which he is playing now. To advance this level, he has to kill all the **N** dragons that are present on this level. IronMan and the dragons have strength, which is represented by an integer. In the fight between him and dragon, the fight's outcome is determined by their strength. Initially, IronMan's strength equals **S**.

If IronMan starts dueling with the **ith (1 <= i <= N)** dragon and IronMan's strength is not greater than the dragon's strength **Xi**, then IronMan loses the duel and dies. But if IronMan's strength is greater than the dragon's strength, then he defeats the dragon and gets a bonus strength increase by **Yi**.

**IronMan can fight the dragons in any order**. Your task is to find out if he will be able to win all the duels without any loss.

**Input Format:**  
The first line will contain two space-separated integers, the initial strength S and the number of dragons N.  
Then N lines follow, each of which contains two space-separated integers **Xi** and **Yi**, where **Xi** is the strength of the dragon and **Yi** is the bonus strength which IronMan gains after defeating this dragon.

**Output Format:**  
Print a single line containing either "YES" (without quotes), if IronMan can defeat all dragons or "NO" (without quotes) if he can't.

**For example:**  
**Input**:  
2 2  
1 99  
100 0

**Output**:  
YES

**Explanation**:  
IronMan's initial strength is 2 and the number of dragons also equals 2  
The first dragon has strength equal to 1 and thus IronMan can defeat this dragon and gain 99 points  
making his strength equal to 2 + 99 = 101.  
Since 101 > 100, IronMan can defeat this dragon also and thus the answer is YES.

**Second example:**

**Input:**  
8 1  
100 100

**Output:**  
NO

**Explanation**:  
IronMan's initial strength is 8 but the required strength for the first dragon is 100 and hence, he dies and we print "NO".

ANS.

**current\_strength,number\_of\_dragons=map(int,input().split())**

**dragons=[]**

**for i in range (number\_of\_dragons):**

**dragon\_strength,bonus=map(int,input().split())**

**dragons.append((dragon\_strength,bonus))**

**dragons.sort()**

**while (len(dragons)>0):**

**fighter\_dragon=dragons.pop(0)**

**if (current\_strength<=fighter\_dragon[0]):**

**print("NO")**

**break;**

**else:**

**current\_strength+=fighter\_dragon[1]**

**else:**

**print("YES")**

#### **Number inversions**

Rahul is facing a unique problem, which he doesn't know how to solve. The problem asks him to build the smallest possible number by applying inversion on any digit of the number any number of times.  
Inversion of a digit is defined as that digit being replaced by 9 minus that digit. Meaning that inversion of 9 will be 9 - 9 = 0 and inversion of 1 will be 9 - 1 = 8 and so on.

The final output should not have any leading zeroes.

**Input Format:**  
The only line of the input contains an integer N

**Output Format:**  
Print only one single integer on a line as described above.

**Constraints:**  
1 <= N <= 10^18

**Examples:**  
**Input:**  
87  
**Output:**  
12

**Explanation:**

9-8 = 1

and 9 -7 = 2

if we see carefully, 12 is the smallest possible number we can get.

**Example:**  
**Input:**  
99  
**Output:**  
90

**Explanation:**

In this case, we do not replace the first 9 since it will lead to a leading zero.

ANS.

1.MY OWN LOGIC:

**n=int(input())**

**m=str(n)**

**l=[]**

**for i in range(1,len(m)):**

**a=int(m[i])**

**b=9-a**

**if(b>=a):**

**l.append(str(a))**

**else:**

**l.append(str(b))**

**if (m[0]=='9'):**

**l.insert(0,'9')**

**else:**

**c=9-int(m[0])**

**if(c>=int(m[0])):**

**l.insert(0,m[0])**

**else:**

**l.insert(0,str(c))**

**string=""**

**for elm in l:**

**string+=elm**

**print(string)**

2. CREATOR OF PROB’S LOGIC:

**n=str(input())**

**number = [char for char in n]**

**for i in range(len(number)):**

**digit=int(number[i])**

**possible\_digit=9-digit**

**if(possible\_digit<digit):**

**if(i==0 and possible\_digit==0):**

**continue**

**number[i]=str(possible\_digit)**

**print("".join(number))**

3. SHREYASH’S LOGIC:

**n=input()**

**inv=""**

**if n[0]=='9':**

**inv=n[0]**

**else:**

**temp=str(9-int(n[0]))**

**if temp<n[0]:**

**inv+=temp**

**else:**

**inv+=n[0]**

**for i in n[1:]:**

**temp=str(9-int(i))**

**if temp<i:**

**inv+=temp**

**else:**

**inv+=i**

**print(inv)**

#### **Rotating Words**

After solving the previous problem, Kartik is stuck on yet another new problem given to him by his teacher.  
The problem wants him to find out the number of unique words that can be formed by cyclically rotating a given string.

Cyclic rotation of a string is defined as moving the last character of the string on the first position.

Kartik can apply Cyclic rotation any number of times.

You have to help him in finding the maximum number of unique words that can be formed by applying Cyclic rotation any number of times.

**Input Format:**  
The only line of the input contains a string S containing of lowercase characters

**Output Format:**  
Print only one single integer denoting the number of unique words that can be formed as explained above.

**Constraints:**  
1 <= len(s) <= 50

**Example:**  
**Input:**  
abcd  
**Output:**  
4  
**Explanation:**  
abcd,dabc,cdab,bcda  
4 unique strings are possible

**Input:**  
cc  
**Output:**  
1  
**Explanation:**  
Only 1 unique string is possible , 'cc'

**ANS.**

**s=input()**

**o=s**

**n=1**

**while True:**

**s=s[1:]+s[0]**

**if o==s:**

**break**

**else:**

**n+=1**

**print(n)**

**TIP: JAHAN BHI LAGE EK VARIABLE LOOP MAIN LENE SE PROB SOLVE NAHI HOGA TOH WHILE LOOP LO**

#### **Picnic Groups**

After solving yesterday's problem, Adarsh is now enjoying and decides to go on a picnic.  
All of his classmates are divided into 'N' groups. Cabs are going to be ordered depending on the number of students.  
The classmates which belong to the same group have to travel together. Each cab can carry at most 4 students. What is the minimum   
the number of cabs needed if all classmates of each group should travel in the same car and one car can take more than one group.

**Input Format:**  
The first line of the input contains N, the number of groups  
The second line contains N space-separated integers, ith of them denoting the size of the ith group

**Output Format:**  
Print one line containing a single integer denoting the minimum number of cabs required.

**Constraints:**  
**1 <= N <= 50  
1 <= size of each group <= 4**

**Examples:**  
**Input:**  
5  
1 2 4 3 3  
**Output:**  
4

one cab will be required by the group having 4 people  
one cab will be required by a group of people having 3 and 1 person(s).  
Now we are left with two groups of size 2 and 3.  
Both of them cannot travel in the same cab because the size will become 3 + 2 = 5, and since each cab can carry only 4 people at max  
we will be requiring a new cab for both these groups. Hence the minimum number is 4.

**ANS.**

**n=int(input())**

**l=list(map(int,input().split()[:n]))**

**m=sorted(l)**

**taxi=0**

**while(len(m)>1):**

**s=m[0]+m[-1]**

**if(s>4):**

**m.pop(-1)**

**taxi+=1**

**elif(s==4):**

**m.pop(0)**

**m.pop(-1)**

**taxi+=1**

**elif(s<4):**

**m.append(m.pop(0)+m.pop(-1))**

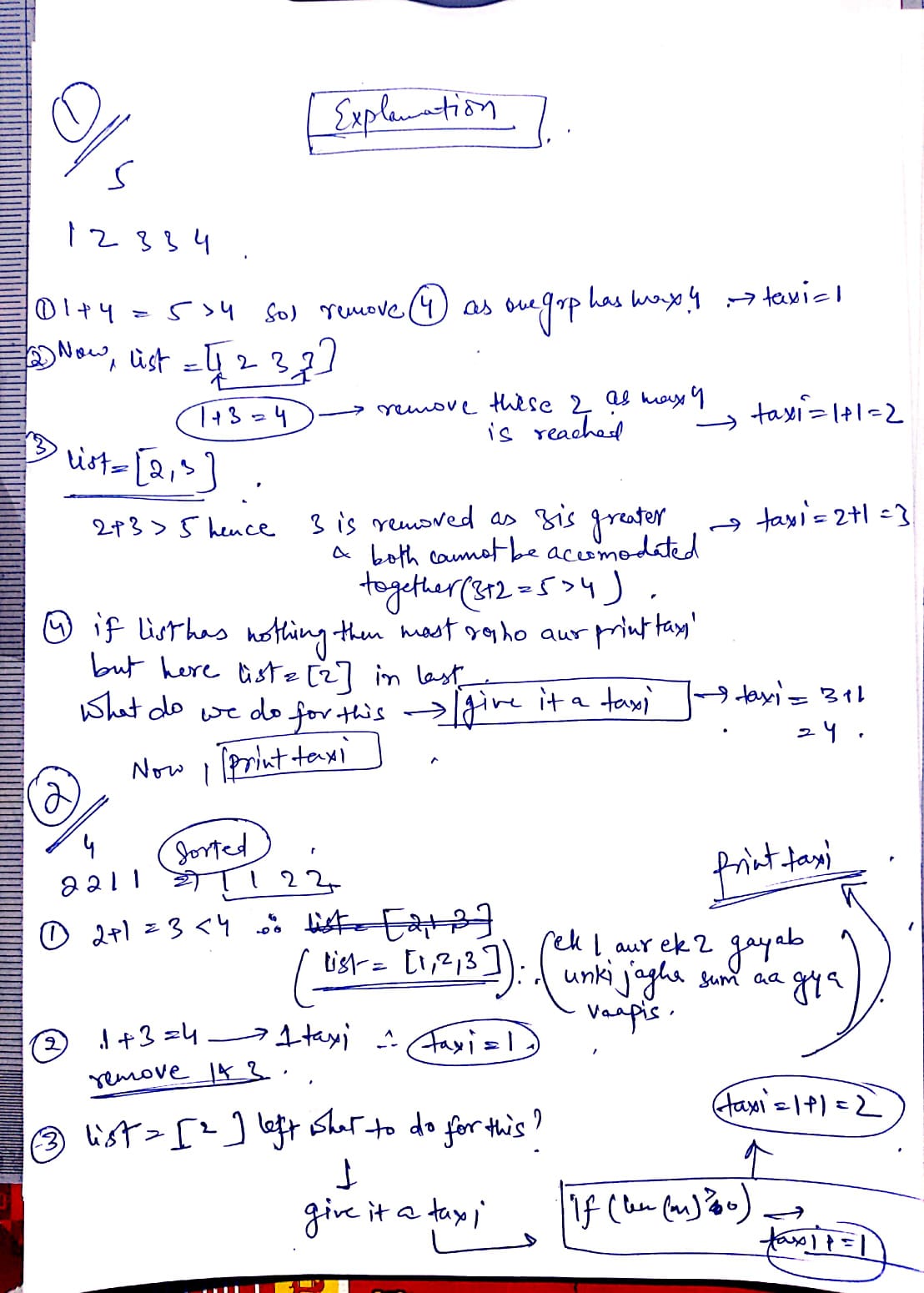
**if (len(m)>0):**

**taxi+=1**

**print(taxi)**

**EXPLANATION:**

**NEXT PAGE:**

****

Fibanocci Increment Mixed String

Increment mixed string is an operation which operates on two strings S1 and S2 of same length to generate a new string S3. The letters in odd position of S3 is one more than the corressponding letter in S1 and the letters in even position of S3 is one more than the corressponding letter in S2. For example, if S1 = ‘amey’ and S2 = ‘dhft’ then S3 = ‘bifu’. For letter ‘z’, letter ‘a’ is one more than it.

Fibanocci increment mixed string is operation which operates on the last two strings in the series. Given two strings, S1 and S2 write a code to generate the nth element using Fibanocci increment mixed string operation. The given strings S1 and S2 are the first two elements in the Fibanocci increment mixed string series. Third element in the series is found by applying increment mixed string operation for first two elements.

If S1 = ‘amey’ and S2 = ‘dhft’ then the first five elements in the series are as follows:

amey

dhft

bifu

ejgv

ckgw

**Input Format**

First line contains the string S1

Next line contains the string S2

Next line contains the value of n

**Output Format**

Print the nth element in the Fibanocci increment mixed string series

**ANS.**

**def incmix (s1,s2):**

**l=[]**

**for i in range(0,len(s1)):**

**if(i%2!=0):**

**if(s2[i]!='z'):**

**l.append(chr(ord(s2[i])+1))**

**else:**

**l.append('a')**

**else:**

**if(s1[i]!='z'):**

**l.append(chr(ord(s1[i])+1))**

**else:**

**l.append('a')**

**string=""**

**for elm in l:**

**string+=elm**

**return string**

**a=str(input())**

**b=str(input())**

**n=int(input())**

**s=[a,b]**

**i=0**

**while(i<n-2):**

**s.append(incmix(s[0],s[1]))**

**s.pop(0)**

**i+=1**

**print(s[1])**

**SHREYASH CODE:**

**s1=input()**

**s2=input()**

**n=int(input())**

**for i in range(n-2):**

**s3=""**

**for i in range(len(s1)):**

**c1=s1[i]**

**c2=s2[i]**

**if (i+1)%2==0:**

**if c2=='z':**

**s3+='a'**

**else:**

**s3+=chr(ord(c2)+1)**

**else:**

**if c1=='z':**

**s3+='a'**

**else:**

**s3+=chr(ord(c1)+1)**

**s1,s2=s2,s3**

**print(s2)**

#### **Letter Partners Game**

In a fun game, every letter in English alphabet has a partner. The first thirteen letters of the English alphabet are called pre-partners and the next thirteen letters are called post-partners. That is ‘a’ is the pre-partner and the corressponding post-partner is ‘n’, ‘b’ is the pre-partner and the corressponding post partner is ‘o’ ....‘m’ is the pre-partner and ‘z’ is the post-partner.

In this game, players will be asked to take a lot with a string ‘w’ and they are said to won the game if the following conditions are satisfied by the letters in ‘w’:

(i) The string may be mixed with pre-partners and post-partners but all pre-partners should have a post-partner

(ii) A pre-partner should come before it’s corressponding post-partner

(iii) For a pre-partner that is in position ‘i’ it’s post-partner

(a) Shall come immediately at posititon ‘i+1’

                                                      or

(b) Should come before all corressponding post-partners of pre-partners that is in positions < i and after all corressponding post-partners of pre-partners that is in position > i.

And the player has lost the game otherwise.

For example, if the word in the lot taken is ‘abon’ then the player has won the game. All pre-partners come before the post-partner and condition (iii) is also satisfied as follows:

1) ‘o’ comes immediately after its pre-partner ‘b’, and as per condition (a) of (iii) it is acceptable

2) ‘n’ comes after its prepartner ‘a’ and it comes after the post-partners of pre-partners that has come after ‘a’ and as per condition (b) of (iii) it is acceptable

Whereas if the word in the lot taken is ‘abno’ then the player has lost the game as the post-partener ‘n’ for pre-partner ‘a’ has come before the post-partner of the pre-partner ‘b’, violates condition (iii).

And if the word in the lot is ‘aerfsbon’ then also the player has won the game as shown in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pre-Partner** | **Post-Partner** | **Right / Wrong** | **By Condition Number** |
| a | n | Right | (i), (ii), (iii) b |
| e | r | Right | (i), (ii), (iii) a |
| f | s | Right | (i), (ii), (iii) a |
| b | o | Right | (i), (ii), (iii) b |

**Input Format**

First line contains the word, w

**Output Format**

Print either Won or Lost

**ANS.**

**w=input()**

**m=list(w)**

**l1=[]**

**l2=[]**

**for i in range(97,110):**

**l1.append(chr(i))**

**for i in range(110,123):**

**l2.append(chr(i))**

**if m[0] in l2:**

**print("Lost")**

**l3=[]**

**for i in range(len(m)):**

**if m[i] in l1:**

**if m[i+1] in l2:**

**if (l1.index(m[i])==l2.index(m[i+1])):**

**l3.append(m[i])**

**l3.append(m[i+1])**

**else:**

**flag=1**

**l4=sorted(list(set(m)-set(l3)))**

**if l4[0] in l1:**

**if l4[1] in l2:**

**if(l1.index(l4[0])==l2.index(l4[1])):**

**print("Won")**

**else:**

**print("Lost")**

**if(flag==1):**

**print("Lost")**

**SHREYASH CODE:**

**w=input()**

**result="Won"**

**if ord(w[0])>109:**

**result="Lost"**

**print(result)**

**m=w**

**if result=="Won":**

**occ=0**

**length=0**

**while True:**

**j=0**

**for i in w[:-1]:**

**ch=i**

**ch1=m[j+1]**

**if ord(ch)<110 and ord(ch)+13==ord(ch1):**

**w = w[:j] + w[(j+2):]**

**j+=1**

**m=w**

**if len(w)==0:**

**break**

**elif(len(w)==length and occ>0):**

**result="Lost"**

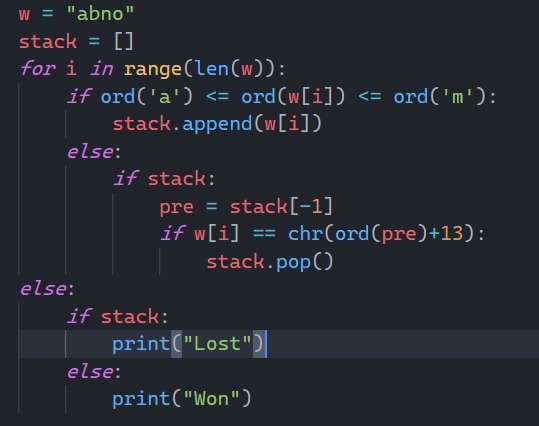
**break**

**length=len(w)**

**occ+=1**

**print(result)**

**MONISH CODE:**

****

**YEH SIRF ABNO KE LIYE WAISE INPUT LO STRING KO**

#### **Image Transformation**

Digital images are represented as a matrix and each element in the matrix represents the RGB value of a pixel. Given the pixel values of a nXn image. Write a code to transform the image into another image by alternate swap of edge pixels. That is for example, given a 4 X 4 image as follows:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

The edge elements of the above image are 1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9 and 5. The following steps are involved in the transformation:

1 and 3 are swapped

1 and 8 are swapped

1 and 16 are swapped

1 and 14 are swapped

1 and 9 are swapped

After doing alternate swap of edge elements the pixel values of the image looks as follows:

3 2 8 4

5 6 7 16

1 10 11 12

13 9 15 14

And given an image of dimension 5 X 5 as follows:

2 4 5 6 3

3 11 12 7 9

7 15 13 8 1

2 21 25 13 14

17 10 16 19 3

Should transform as:

5 4 3 6 1

3 11 12 7 9

2 15 13 8 3

2 21 25 13 14

7 10 17 19 16

**Input Format**

First line contains the dimension of the matrix, n

Next nxn lines contain the elements of the matrix in row major order. That is the elements in the first row are given followed by second row and so on

**Output Format**

Print the transformed nxn matrix in ‘n’ lines

Each line contains the elements of a row separated by a tab.

**Note:**There is a tab at the end of each line

**ANS.**

**SHREYASH AND MINE LOGIC: (LEARN HOW TO PUT BACK THE ELEMENTS IN THE MATRIX)**

**n=int(input())**

**m=[[int(input()) for x in range(n)] for y in range(n)] #HOW TO INPUT MATRIX**

**#HOW TO APPEND THE MATRIX ELEMENTS IN A LIST :**

**l=[]**

**for j in range(n):**

**l.append(m[0][j])**

**for i in range(1,n-1):**

**l.append(m[i][n-1])**

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

**l.append(m[n-1][j])**

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

**l.append(m[i][0])**

**for i in range(0,len(l)-2,2):**

**l[i],l[i+2]=l[i+2],l[i]**

**#HOW TO PUT ELEMENTS BACK IN THE MATRIX**

**k=0**

**for j in range(n):**

**m[0][j]=l[k]**

**k+=1**

**for i in range(1,n-1):**

**m[i][n-1]=l[k]**

**k+=1**

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

**m[n-1][j]=l[k]**

**k+=1**

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

**m[i][0]=l[k]**

**k+=1**

**#HOW TO PRINT MATRIX**

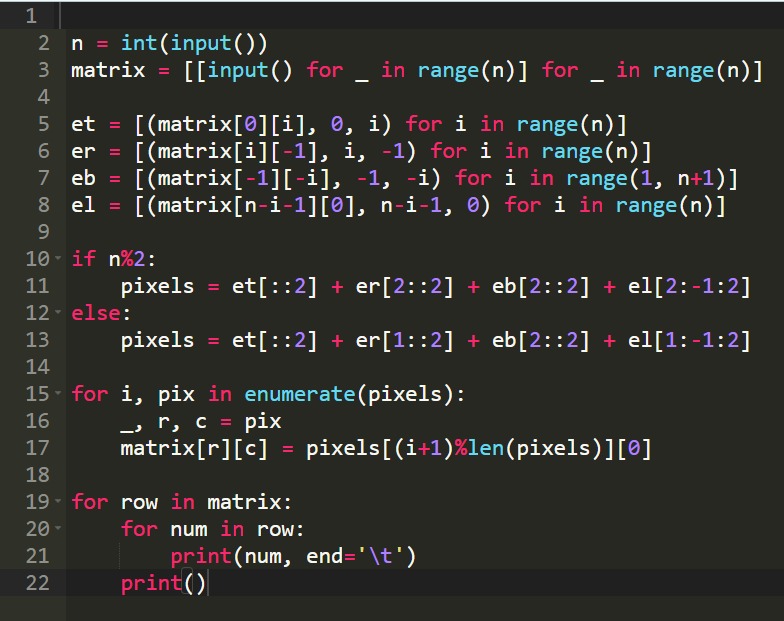
**for i in range(n):**

**for j in range(n):**

**print(m[i][j], end = "\t")**

**print()**

**MONISH CODE:**

****

#### **Minor DNAs**

**Deoxyribonucleic acid (DNA)** is a molecule composed of two chains that coil around each other and carry genetic instructions used in the growth, development, functioning and reproduction of all living organisms. The DNA is composed of simpler units called nucleotides. Each nucleotide is composed of one of four nitrogen-containing nucleobases cytosine [C], guanine [G], adenine [A] or thymine [T]. There are specific rules by which a DNA is formed with these four nucleobases. Example DNAs are:

ccag

actaatagtacccataaacctgcta

Many research is being carried out based on the occurrences of the nucleotides in a DNA.

Here a minor DNA ‘d1’ of length ‘k’ can be obtained from a given DNA ‘d’ of length ‘n’ by deleting ‘n-k’ nucleotides in ‘d’. For example, if the given DNA ‘d’ of length four is ‘ccag’ then all the possible minor DNAs of length three are four minor DNAs:

Remove last letter - cca

Remove third letter - ccg

Remove second letter - cag

Remove first letter – cag

Minor DNAs of length 4 of a DNA ctctac which is of length 6 are:

ctct, ctca, ctta, ccta, ctcc, cttc, cctc, ctac, ccac, ctac, tcta, tctc, tcac, ttac and ctac.

Given a DNA of length ‘n’ and a length ‘k’ of minor DNAs required, write a code to generate all possible minor DNAs of length ‘k’ and print them in ascending order.

**Input Format**

First line contains the DNA, d

Next line contains the length of the minor DNAs required, k

**Output Format**

Print all minor DNAs of length ‘k’ in ascending order

Print one minor DNA in one line

**ANS.**

**dna=input()**

**n=int(input())**

**d=list(dna)**

**from itertools import combinations**

**comb=list(combinations(d,n))**

**l=[]**

**for i in comb:**

**string=""**

**for j in i:**

**string+=j**

**l.append(string)**

**l.sort()**

**for i in l:**

**print(i)**

#### **Win in a Board Game**

In a board game, a pXp board is given and the cells of the board are numbered as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| 3,0 | 3,1 | 3,2 | 3,3 |
| 2,0 | 2,1 | 2,2 | 2,3 |
| 1,0 | 1,1 | 1,2 | 1,3 |
| 0,0 | 0,1 | 0,2 | 0,3 |

The movement of the coin in the board allowed are left, right, up and down. The coin movements are defined in the below table:

|  |  |  |
| --- | --- | --- |
| **Movement** | **Current Position** | **Position after move** |
| Left | m, n | m, n-1 if n>0  m, n otherwise |
| Right | m, n | m, n+1 if n<p-1  m, n otherwise |
| Up | m, n | m+1, n if m<p-1  m, n otherwise |
| Down | m, n | m-1, n if m>0  m, n otherwise |

Given a size the board, p, initial cell of the coin in the board, a list of movements and a winning cell in the board, write a code to check if the coin reaches the winning cell from initial cell by these movements. If the winning position is reached then print Win and print Loss otherwise.

**Input Format**

Dimension of the board, p

Initial cell of the coin m,n separated by a space

Number of moves, num1

Next num1 lines contain the movements l-left, r-right, u-up and d-down

Winning cell of the coin m1, n1 separated by a space

**Output Format**

Print Win or Loss

**ANS.**

**p = int(input())**

**m,n = map(int,input().split())**

**num1 = int(input())**

**moves = [input() for i in range(num1)]**

**m1,n1 = map(int,input().split())**

**for move in moves:**

**if move == 'l' and n > 0: n-=1**

**elif move == 'r' and n < p-1: n+=1**

**elif move == 'u' and m < p-1: m+=1**

**elif move == 'd' and m > 0: m-=1**

**if m == m1 and n == n1: print("Win")**

**else: print("Loss")**

#### **Find the Lucky Kids**

In a game for kids under age seven, m\*n kids are made to sit in a room where there are ‘n’ chairs arranged in ‘m’ rows. The game organizers has got ‘m\*n’ chits in which either one of the two numbers ‘p’ or ‘q’ is written. Each kid is asked to pick a chit. A lot is taken by the coordinator which contains the lucky number ‘l’ (Value of 'l' will be either 'p' or 'q'). Then it is declared that “Chit Transition” takes place for few kids. If the lucky number ‘l’ is equal to ‘p’, during chit transition, a kid ‘k’, sitting in the row ‘r’ and column ‘c’ with number ‘p’ is given ‘q’ when one of the following criteria is satisfied:

(i) If atleast one of the kid sitting in the row ‘r’ has ‘q’

(ii) If atleast one of the kid sitting in the column ‘c’ has ‘q’

After this Chit transition phase, the kids with number ‘p’ are declared as winners. For example, if there are 3\*3 kids, chit numbers as 3 and 5 seated with original chit number as follows:

3 5 3

3 3 3

3 5 5

If the lucky number taken by coordinator is 3 then the chit number of the kids after “Chit Transition” phase will be:

5 5 5

3 5 5

5 5 5

Now the kid sitting in the second row and first column is the winner. Sometimes there can be more than one winner in this game and sometimes there may not be any winner in this game also. Print the row and column position of all the winning kids and when there are no winners print ‘No winner’.

**Input Format**

First line contains the number, m

Next line contains the number, n

Next line contain the number, p

Next line contain the number, q

Next ‘m’ lines contain the chit number of ‘n’ kids seated on each row separated by a space

Next line contains the lucky number, l

**Output Format**

Print the row and column position of winning kids in order. Print the position sorted row wise, when row of more than one position is same print them sorted column wise and Print ‘No winner’ when no one has won the game

ANS.

m=int(input())

n=int(input())

p=int(input())

q=int(input())

matrix=[]

for i in range(m):

row=list(map(int,input().strip().split()))[:n]

matrix.append(row)

l=int(input())

flag=0

for i in range(m):

for j in range(n):

if matrix[i][j]==l:

a=[]

b=[]

for x in range(m):

a.append(matrix[x][j])

for y in range(n):

b.append(matrix[i][y])

if l==p:

if q not in list(a):

if q not in list(b):

flag=1

print(i+1,j+1)

if l==q:

if p not in list(a):

if p not in list(b):

flag=1

print(i+1,j+1)

if flag==0:

print("No winner")

#### **Ram and Pipes**

Ram is a plumber who is working at a construction site where he has to build the entire piping system of the site. He has small pipes of various lengths which he can use to build a long pipe. However, whenever he wants to join two pipes there is an associated cost due to welding, raw material etc. The cost of joining two pipes is equal to the lengths of the two pipes being joined.

Ram really has a tight budget and wants your help in minimizing the overall cost of combining all the pipes.

**Input format:**

The first line contains a single integer ‘N’ denoting the number of pipes available.

Next N lines each contain a single integer denoting the length (L) , of the ith pipe

**Output format:**

Print a single line containing 1 integer, the minimum cost to join all pipes

**Constrains:**

1 <= N <= 500000

1 <= L <= 500

**Examples:**

**Sample input 1:**

2

1

2

**Sample output 1:**

3

**Explanation**

The minimum cost to join these pipes will be 1 + 2 = 3

**Sample Input 2:**

3

1

2

3

**Sample Output 2:**

9

**Explanation:**

We combine 1 and 2, and get a pipe of length 3 with a cost of 3

Now, we have two pipes of length 3 each, so the cost of joining these two

Pipes will be 3 + 3 = 6

So total cost = 6 + 3 = 9.

**Hint:**If you think that your logic is correct and you’re still getting TLE, then you should consider searching online for some data structure implemented by a python module (heapq or PriorityQueue) and understand how they can help you in solving this problem efficiently with the given constraints. You will get to know more in your Data Structures & Algorithms course, but if you are able solve this question now, excellent!

ANS.

from queue import PriorityQueue

n=int(input())

pipes=PriorityQueue()

for i in range(n):

pipes.put(int(input()))

total\_cost=0

while (pipes.qsize()>1):

pipe1=pipes.get()

pipe2=pipes.get()

total\_cost+=(pipe1+pipe2)

pipes.put(pipe1+pipe2)

if n>1:

print(total\_cost)

else:

print(pipes.get())

OLD GHISA PITA LOGIC(MAYBE USEFUL WHEN I DO C++):

n=int(input())

l=[]

for i in range(n):

e=int(input())

l.append(e)

m=sorted(l)

if (n!=1):

k=[]

while(len(m)!=1):

k.append(m[0]+m[1])

m.append(m[0]+m[1])

m.pop(0)

m.pop(0)

m.sort()

print(sum(k))

else:

print(m[0])

#### **LinkedIn Connections**

Whenever two people are connected to each other on linkedin, they are considered as connections of each other. A term **degree**of connection is defined by the distance between a person and the connection. For example, if Rahul is connected to Neha and Neha is connected to Raj, in this case:

Connection: Degree

Neha-Raj: 1

Rahul-Neha: 1

Rahul-Raj: 2

Raj-Neha: 1

It is very important to grow your network on LinkedIn, and so in order to do this, one needs to send connection requests to different people. However, LinkedIn has a restriction in terms of a person to whom you can send a connection request. You can only send connection requests to people who are your 3rd degree connections, meaning that the distance between you and that person should be **at most 3**. Given the number of people on the linkedin platform and all the connections, answer n queries in which ith (1<= i <= n) query asks you if a connection request can be sent to the person or not.

**Input Format:**

First line of the input contains a single integer, the number of people: **N**

Second line of the input contains a single integer the number of connections: **M**

Next M lines contain two integers each, separated by space: (Xi Yi) ; 1 <= i <= M

Next line contains an integer, person who wants to send connection requests: **K**(1 <= K <= N)

Next line contains, one single integer, the number of queries, **Q**

Next Q lines, each containing an integer, jth (1 <= j <= Q) containing an integer **L**(1 <= L <= N)

**Output Format:**

Output **Q**lines each containing a string as follows:

“Yes” (without quotes), if the connection request can be sent.

“No” (without quotes), if the connection request cannot be sent.

**Constraints:**

1 <= 50 <= N

1 <= 50 <= M

1 <= 50 <= Q

**Example:**

**Input:**

12

13

1 2

1 5

2 4

5 3

5 6

1 7

7 8

1 9

9 10

7 10

10 11

11 12

4 6

1

3

2

12

11

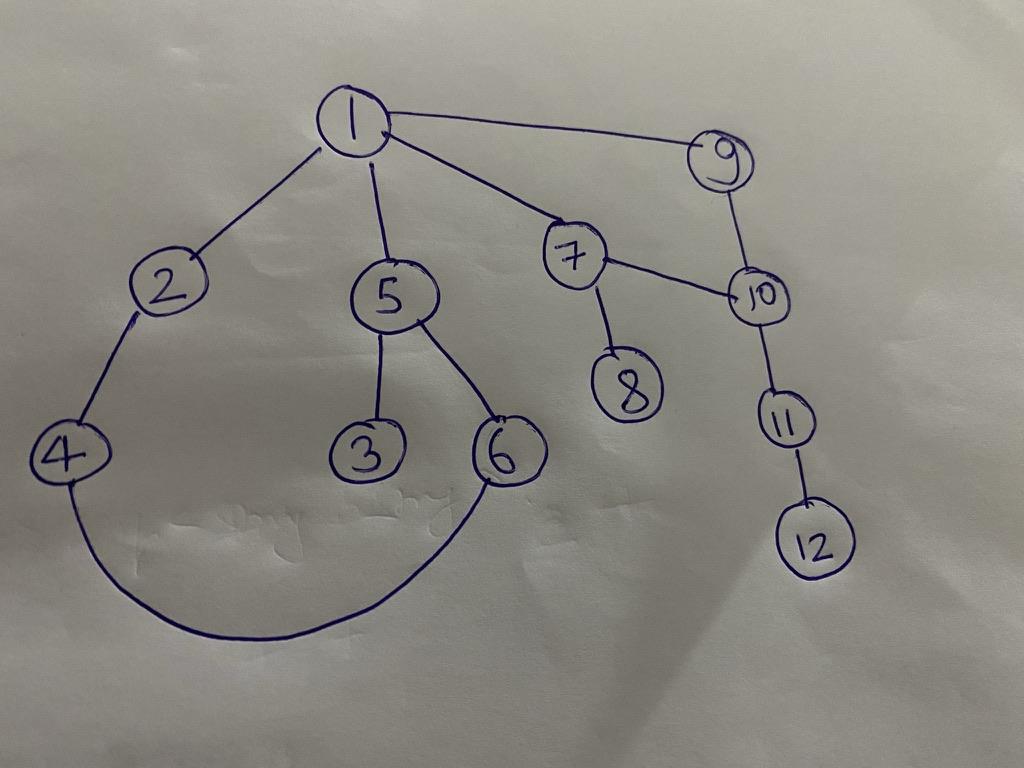
**Output:**

yes

no

yes

**Explanation:**



As can be seen from this network of connections, the answers can be given appropriately  
For 1 and 11 degree is 3, which means the request can be sent, but 12 is a 4th degree connection for 1, so request cannot be sent, and 2 is 1st degree so we print yes.  
**Note:** if a person is already connected, like 2 in this case is already connected to 1, we still print “yes”.

ANS.

INF=10000000000

n=int(input())

m=int(input())

network=[]

for i in range (n+1):

network.append([])

for i in range(m):

x,y=map(int,input().split())

network[x].append(y)

network[y].append(x)

source = int(input())

visited = [False] \* (n+1)

distance = [INF] \* (n+1)

current\_nodes = []

distance[source] = 0

current\_nodes.append (source)

visited[source]=True

while(len(current\_nodes)>0):

node = current\_nodes.pop(0)

for neighbour in network[node]:

if(visited[neighbour] == False):

visited[neighbour] = True

distance[neighbour] = distance[node] + 1

current\_nodes.append(neighbour)

queries=int(input())

for i in range (queries):

k = int(input())

if(distance [k] > 3):

print("no")

else:

print("yes")

SHREYASH’S CODE: THIS WORKS IN VPROPEL:

INF=1000

n=int(input())

m=int(input())

network=[]

for i in range (n+1):

network.append([])

for i in range(m):

x,y=map(int,input().split())

if(x>n):

network[n].append(y)

else:

network[x].append(y)

if(y>n):

network[n].append(x)

else:

network[y].append(x)

source = int(input())

visited = [False] \* (n+1)

distance = [INF] \* (n+1)

current\_nodes = []

distance[source] = 0

current\_nodes.append (source)

visited[source]=True

while(len(current\_nodes) > 0):

node = current\_nodes.pop(0)

for neighbour in network[node]:

if(visited[neighbour] == False):

visited[neighbour] = True

distance[neighbour] = distance[node] + 1

current\_nodes.append(neighbour)

queries=int(input())

for i in range (queries):

k = int(input())

if(distance[k] > 3):

print("no")

else:

print("yes")

#### **Taxonomy Search**

In linguistics, **hyponym** and hypernym are a type-of relationship taxonomy among the words or phrases based on their meaning. A word ‘W2’ is said to be hyponym of another word ‘W1’ if the meaning of ‘W2’ is included within ‘W1’. If ‘W2’ is hyponym of ‘W1’ then ‘W1’ is said to be hypernym of ‘W2’.

For example, cabbage and spinach are hyponym of greens, peas and beans are hyponyms of pulses, carrots and turnips are hyponym of roots, and potatoes and yams are hyponyms of tubers. Greens, pulses, roots and tubers are inturn hyponyms of vegetables. The taxonomy can be viewed in the following url.

<https://tinyurl.com/yd7se5pv>

In the above taxonomy, vegetables is at level 0, greens, pulses, roots and tubers are at level 1 and cabbage, spinach, peas, beans, carrots, turnips, potatoes and yam are at level 2.

Another example taxonomy can be seen at url:

<https://tinyurl.com/y88ywkbb>

Level 0 – Color

Level 1 – Purple, Red, Blue, Green

Level 2 of Purple – Crimson, Violet, Lavender

Given ‘n’ taxonomies of maximum level as 2, a name ‘p’ at level 0, a name ‘l1’ at level 1 and a name ‘c’ level 2 check p, l1 and c are at proper level of the same taxonomy.

**Input Format**

First line contains the number of taxonomies, n

Next few lines contains details of each taxonomy in the following order

Name of the element in the level 1 of the taxonomy

Next line contains some number of elements in level 1 of the taxonomy separated by a space

Next few lines contains the details of elements in level2 of the taxonomy separated by a space, the first string here is the name of the element in level1 and remain strings are the children of the element in level1

If no element is present at level2 for an element in level1 then None is given

Next line contains the name of the element to be checked at level 0, p

Next line contains the name of the element to be checked at level 1, l1

Next line contains the name of the element to be checked at level 2, l2

**Output Format**

Print Taxonomy present if elements p,l1 and c belong to the same taxonomy at appropriate levels and Taxonomy not present otherwise

**Illustration of Input**

The **vegetable taxonomy** is represented as follows:

vegetables

greens pulses roots tubers

pulses peas beans

roots carrots turnips

greens cabbage spinach

tubers potatoes yams

**Color taxonomy is represented as:**

color

purple red blue green

purple crimson violet lavender

None

None

None

ANS.

SHREYASH’S CODE:

n=int(input())

l={}

for i in range(n):

l0=input()

l.update({l0:{}})

l1=input().split()

for j in l1:

l[l0].update({j:{}})

for k in range(len(l1)):

l2=input().split()

for m in l2[1:]:

l[l0][l2[0]].update({m:{}})

flag = True

while True:

try :

i=input()

if i not in l.keys():

flag=False

else:

l=l[i]

except:

if flag==False:

print("Taxonomy not present")

break

else:

print("Taxonomy present")

break

#### **Kartik and Typing**

Kartik has been given many words to type. For each word he types he consumes some time depending on the character that is going to be pressed.  
For simplicity, we are going to assume that all the words he has to type consist of only 4 letters 'd' , 'f', 'j' and 'k'.  
He can type 'd' and 'f' with the left hand and the letters 'j' and 'k' using the right hand.  
When typing a word he takes, he takes 2 seconds to type the first character, for each of the other character he types, he takes 2 seconds to type if the character uses other hand  
and 4 seconds if the character uses the same hand.  
If he has typed a word already then he can type that word in half the time taken to type it for the first time.  
Given some words, calculate the total time taken to type all the given words.

**Input:**  
The first line of input contains 'N', the number of words to type.  
Subsequence N lines each contain a string denoting the word that Kartik has to type.

**Output:**  
Print one single line containing the time taken to type all the words.

**Example:**  
**Input:**  
4  
dfjk  
ddd  
jkd  
dfjk

**Output:**  
36

ANS.

x=0

l={}

for i in range(int(input())):

a=input()

if a in l:

x+=l[a]/2

continue

d=2

for j in range(1,len(a)):

if(a[j] in 'df' and a[j-1] in 'df' or a[j] in 'jk' and a[j-1] in 'jk'):

d+=4

else:

d+=2

x+=d

l[a]=d

print(int(x))

#### **Points in ICC World Test Championship**

The ICC World Test Championship is a league competition for Test cricket run by the International Cricket Council (ICC), which started on 1 August 2019. The result of a test match could be win, lose, draw or tie. According to the ICC Test Championship, each of the nine top-ranked sides will play six series over two years. Totally 27 Test series will be played.

Each series can have number of matches from 2 to 5. Each country can choose the number of matches that they will play in each series. Irrespective of the number of matches in the series, total points of each series is 120 and the 120 points is equally divided between the matches in the series.

Hence, in a two-match series the winner of a match will get 60 points; while in a four-match series, the winner of a match will get 30 points. In case of a match being tie, half of the total points available for the match will be offered to either side, while for a match that ends up in draw, one-third of the available points of the match will be given to both sides.

Given the number of points scored by both the teams till now, calculate the total number of points of the teams in a scenario, given name of a team which wins the next two matches and the last match of the series ends up in a draw. Check the validity of the team name and print ‘Invalid input’ if the country name is not valid.

**Example**

Currently India vs Australia are playing a four match test series in Australia, and after the loss of Adelaide test India has got 360 points and Australia has got 326 points.When Australia has won the next two matches the total number of points scored by the teams will be:

India = 360+10 = 370

Australia = 326+30+30+10=396

**Input format**

First line will have current points of India

Next line will have current points of Australia

Next line contains the name of the team that has won the next two matches (‘IND for India and ‘AUS’ for Australia)

**Output Format**

First line will have final points of India

Second line will have final points of Australia

ANS. import sys

p1=int(input())

p2=int(input())

name=input()

pfi=0

pfa=0

if (name=='IND'):

pfi=p1+70

pfa=p2+10

elif (name=='AUS'):

pfa=p2+70

pfi=p1+10

else:

print('Invalid input')

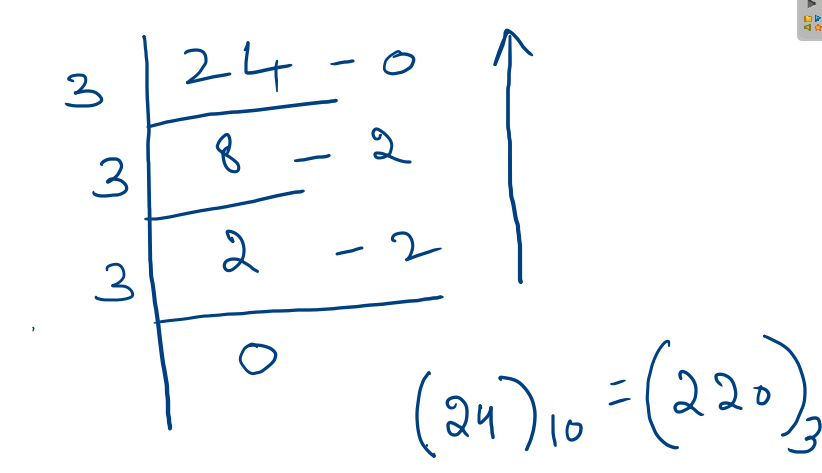
sys.exit()

print(pfi)

print(pfa)

#### **Special Numbers in different bases**

A number in a base is said to be a special number if it is equal to the sum of its own digits each raised to the power of the number of digits. A number in base ‘b’ can have digits from 0 to b-1. By default the numbers used by humans is to base 10 and called as decimal number system. A number ‘n’ from decimal number system can be converted to any base ‘b’ by repeated division of ‘n’ by ‘b’ and writing reminder of each division in reverse order. For example, number 24 is converted to base 3 as shown below:



Given a number ‘n’ and a base ‘b’, write a code to check if n is a special with respect to base ‘b’. If any digit of ‘n’ is greater than ‘b-1’ then print ‘Invalid’

**Example 1:**

**Input**

122

3

**Output**

Yes

**Explanation**

122 -> number of digits - 3

1^3 + 2^3 + 2^3 -> 17

17 to base 10 is 122 to base 3

**Example 2:**

**Input**

433

5

**Output**

Yes

**Example 3:**

**Input**

121

3

**Output**

No

**Example 4:**

**Input**

125

3

**Output**

Invalid

**Input Format**

First line contains the number, n

Next line contains the base of the number,b

**Output Format**

Print Invalid or Yes or No

ANS.

n=(input())

b=int(input())

m=str(n)

x=[]

for i in range(len(m)):

x.append(int(m[i])\*\*len(m))

s=sum(x)

try:

a=int(m,b)

if s==a:

print("Yes")

else:

print("No")

except:

print("Invalid")

SHREYASH’S CODE:

n=input()

b=int(input())

l=(list(map(int,list(n))))

s = sum(list(map(lambda x : x\*\*len(n),l)))

try:

x=int(n,b)

if(s==x):

print("Yes")

else:

print("No")

except:

print("Invalid")

Rohan's Vacation

Rohan was going on a long vacation across various cities in the world. He had made an itinerary containing all the information about his journey. He had flight tickets booked as well.

Unfortunately, at the last minute, someone just ran into him and all his tickets fell on the floor.

Help him to find the correct order of all the tickets by finding the correct path from source to destination.

**Input Format:**

First line of the input contains a number **N,**indicating the number of tickets he has.

Next lines each contain two strings made of alphabets, each separated by a space, the **src**(source) and the **dest**(destination).

**Output format:**

Print a single line containing the entire journey plan (itinery) of Rahul.

**Constraints:**

1 <= N <= 500

1 <= len(src), len(dest) <= 50

Source and destination strings are made up of alphabets(uppercase/lowercase).

**Example Input:**

3

DEL KOL

MAA DEL

KOL GOA

**Example Output:**

MAA DEL KOL GOA

**Explanation:**

Given the tickets from DEL to KOL, MAA to DEL, KOL to GOA,

If we arrange the properly, we can see that Rohan starts from MAA then goes to DEL,

And then from DEL he has ticket to KOL and from KOL he has ticket to GOA, so we print his entire itinerary.

ANS.

l=[]

for i in range(int(input())):

l.append(input().split())

journey=[]

while len(l)>1:

k=0

while k<len(l):

i=l[k]

c=l.copy()

c.remove(i)

flag=0

k+=1

for j in c:

if i[0]==j[1]:

flag=0

break

else:

flag=1

if flag==1:

journey.extend(i)

l.remove(i)

journey.extend(l[0])

i=0

final=[]

for i in journey:

if i not in final:

final.append(i)

print(\*final)

MY CODE:

n=int(input())

a=[]

for i in range(n):

l=[]

x,y=map(str,input().split())

l.append(x)

l.append(y)

a.append(l)

l2=[]

l3=[]

for i in a:

l2.append(i[1])

l3.append(i[0])

q=list(set(l2+l3))

l4=list(set(l3)-set(l2)) #best way to subtract the elements of one list from other list

u=0

p=0

t=0

while(len(l4)+p-t < len(q)+1):

t=p

k=l4[u]

for o in range(len(a)):

if (a[o][0]==k):

l4.append(a[o][1])

u+=1

p+=1

print(\*l4)

#### **Factor Indexed Numbers**

A number ‘n’ in a list at index ‘i’ (human indexing) is said to be a factor indexed number if ‘i’ is a factor ‘n’. Given two lists of numbers l1 and l2 form a list l3 by inserting factor indexed numbers of l1 and l2. Intialize index to ‘0’ visit elements of l1 and l2 at index, if one of them is factored indexed insert and increment index and go till end of the list. If both of them are factor indexed then insert element of l1 at index and then element of l2 at index and increment index. If end of one of the list is reached then add elements of other list to l3 if they are factor indexed. After construction of l3, Print the numbers which continues to be factor indexed in l3.

For example, if l1 contains 15, 12, 17, 32, 26, 42 and l2 contains 45, 34, 64, 80 then l3 will be 15, 45, 12, 34, 32, 80, 42 then print 15, 12, 42.

**Input Format**

First line contains the number of elements in l1, n1

Next ‘n1’ lines contain the elements of l1

Next line contains the number of elements in l2, n2

Next ‘n2’ lines contain the elements of l2

**Output Format**

Print good numbers of l3 one number in a line

ANS.

def fac(lst1,lst2,n):

final=[]

for i in range(n):

if i<len(lst1):

if lst1[i]%(i+1)==0:

final.append(lst1[i])

if i<len(lst2):

if lst2[i]%(i+1)==0:

final.append(lst2[i])

return final

l1=[int(input()) for \_ in range(int(input()))]

l2=[int(input()) for \_ in range(int(input()))]

m=max(len(l1),len(l2))

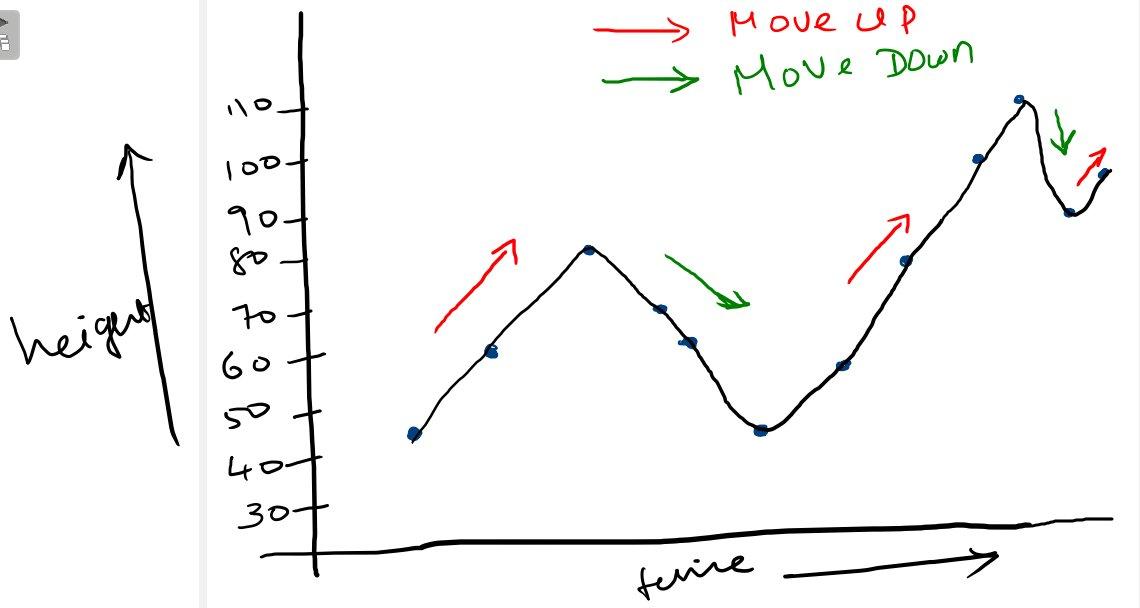
l3=fac(l1,l2,m)

[print(i) for i in fac(l3,[],len(l3))]

#### **Count Move Up and Down**

Ramu makes a travel to one his office in a hilly region by his car. Given the height of his current location above sea level every few minutes after he starts the travel, write a code to check the number of up moves and down moves. First move is always up move.

For example, if 12 points are given as follows 45, 60, 82, 72, 65, 32, 53, 84, 103, 110, 89, 95 then there are 3 up moves and 2 down moves as shown below:



**Input Format**

First line contains the number of current locations of Ramu, n

Next line contains ‘n’ height of locations separated by a space

**Output Format**

Print the number of Up moves and number of Down moves separated by a space

ANS.

**SHREYASH’S CODE:**

**n=int(input())**

**l=list(map(int,input().split()))**

**up=0**

**down=0**

**prev=l[0]**

**c=0**

**d=0**

**for i in l[1:]:**

**if(i>prev):**

**c=1**

**prev=i**

**if(l.index(i)==len(l)-1):**

**up+=1**

**elif(c==1):**

**up+=1**

**c=0**

**if(i<prev):**

**d=1**

**prev=i**

**if(l.index(i)==len(l)-1):**

**down+=1**

**elif(d==1):**

**down+=1**

**d=0**

**print(up,down)**

#### **Best Move for an Industrial Robot**

In an industry, robots are employed to bring specified number of nuts, ‘n’ to its current location. Different numbers of nuts are packed in ‘m’ packages and kept in a rack at a location inside the industry.

Robot can make only one fetch operation. In a single fetch operation, they can fetch packages from one rack or more than one adjacent racks only. Size of the fetch operation is the number of packages taken by the robot. Power consumed by the robot increases with the size of the fetch operation. When fetch size is same, fetching from rack(s) with lesser rack numbers consume less power than fetching from higher rack numbers.

Given the number of nuts required, ‘n’ and the number of nuts in each of ‘m’ packages, write a code to print the number of nuts in the packages that are to be taken by the robot so that it consumes least possible power. If the number of nuts cannot be fetched by the robot print ‘Not possible’.

For example, if the number of nuts required is 10 and the nuts are packed in five pacakages as follows:

2 3 6 4 10

Then 10 is the best possible package of nut(s) that shall be taken. If 18 nuts are required and the nuts are packed in five pacakages as follows:

2 3 6 4 10

then print ‘Not possible’.

**Input Format**

First line contains the number of nuts required, n

Next line contains the number of packages, m

Next line contain the number of nuts in m packages separated by a space

**Output Format**

Print the nuts in the packages that has to fetched and print Not possible if the robot cannot fetch

**ANS.**

**n=int(input())**

**m=int(input())**

**l=list(map(int,input().split()))**

**if n in l:**

**print(n)**

**else:**

**flag=0**

**prev=l.copy()**

**for i in range(len(l)):**

**su=0**

**s=[]**

**for j in range(i,len(l)):**

**s=l[i:j+1]**

**su=sum(s)**

**if su>n:**

**break**

**if su==n and len(prev)>=len(s):**

**flag=1**

**prev=s**

**break**

**if flag==1:**

**print(\*prev)**

**else:**

**print("Not possible")**

Duck and Chicken Eggs

A farmers takes chicken and duck eggs in ‘n’ baskets. In two baskets he has got duck eggs and rest of the bags he has chicken eggs. Number of eggs in each basket is distinct. He tells that number of chicken eggs is twice as that of number of duck eggs. Given the name of the ‘n’ baskets and number of eggs in them, write a code to print the name of baskets with duck eggs and name of basket with chicken eggs. If there can be more than one combination in which duck eggs can be there, consider the first basket name which comes first in the input order

**Input Format**

First line contains the number of baskets, n

Next ‘n’ lines contain the name of the baskets and the number of eggs in them

**Output Format**

Print name of baskets with duck eggs separated by a space in the first line

Print name of baskets with chicken eggs separated by a space in the next line

Print the name of the baskets in the order of their input and there is no space at the end of the lines

**Example1**

**Input**

5

B1 5

B2 12

B3 14

B4 23

B5 6

**Output**

B3 B5

B1 B2 B4

**ANS.**

**MY PROGRAM:**

**n=int(input())**

**baskets=[]**

**eggs=[]**

**for i in range(n):**

**x,y=map(str,input().split(" "))**

**baskets.append(x)**

**eggs.append(int(y))**

**info=dict(zip(baskets,eggs))**

**from itertools import combinations**

**comb=combinations(baskets,2)**

**pairs=list(comb)**

**ducks=[]**

**for i in pairs:**

**sum\_selct\_bkts=info[i[0]]+info[i[1]]**

**sum\_rest\_bkts=sum(eggs)-sum\_selct\_bkts**

**if sum\_rest\_bkts == 2\*(sum\_selct\_bkts):**

**ducks.append(i[0])**

**ducks.append(i[1])**

**break;**

**remaining\_bkts=sorted(list(set(baskets)-set(ducks)))**

**print(\*ducks)**

**print(\*remaining\_bkts)**

#### **Smallest Divisible Number**

Given an integer ‘n’ where n>=1, write a C code to check whether there exist a smallest number ‘S’, where S<=107 and S is divisible by all even numbers ‘m’ where 1<=m<=n.

For example, when n = 10, the smallest number divisible by all even numbers 2, 4, 6, 8 and 10 is 120. If no such number exist then print ‘No such number in range’.

**Input Format**

First line contains the number, n

**Output Format**

Print either the number that is divisible by all even numbers less than ‘n’ or print No such number in range

**ANS.**

**n=int(input())**

**even=[]**

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

**if i%2==0:**

**even.append(i)**

**#STATEMENTS IN YELLOW ARE FOR CALCULATING LCM OF MORE THAN 2 #NOS.**

**from math import gcd**

**lcm = 1**

**for i in even:**

**lcm = lcm\*i//gcd(lcm, i)**

**if (lcm>10\*\*7):**

**print("No such number in range")**

**else:**

**print(lcm)**

#### **Swap If Tail Greater**

To check if a number n1 is tail greater than another number n2, one has to compare the digits from the end of the numbers. When the digits in position ‘i’ are same for the numbers then the previous digits has to be compared till a conclusion can be made. If the elements in position ‘i’ and ‘i+1’ are same then do not swap.

**Input Format**

First line contains the number of elements in the collection, n

Next ‘n’ lines contain the numbers in the collection

**Output Format**

Print the elements after swapping if not in tail greater order mentioned. Print one number in one line

For example, 13 is greater than 41 since 3 the last digit of 13 is greater than 1 which is last digit of 41. 253 is greater than 153 but lesser than 18.

Given a collection of n integers with same number of digits, write a C code to swap adjacent elements if element in position ‘i+1’ is not tail greater than the element in position ‘i’. For example, given five integers 223, 131, 145, 144 and 315 after swap is done elements will be 131, 223, 144, 315, 145.

**ANS. Just swap based on condition no need to sort**

**PYTHON:**

**n = int(input())**

**nums = [int(input()) for i in range(n)]**

**for i in range(n-1):**

**if int(str(nums[i])[::-1]) > int(str(nums[i+1])[::-1]):**

**nums[i],nums[i+1] = nums[i+1],nums[i]**

**for i in nums: print(i)**

**C++**

**#include <iostream>**

**#include <string>**

**#include <algorithm>**

**using namespace std;**

**string rev(string s){**

**int n = s.length();**

**for(int i=0;i<n/2;i++){**

**char temp = s[i];**

**s[i] = s[n-i-1];**

**s[n-i-1] = temp;**

**}**

**return s;**

**}**

**int main(){**

**int t;**

**cin>>t;**

**string nums[t];**

**for(int i=0;i<t;i++) cin>>nums[i];**

**for(int i=0;i<t-1;i++){**

**int u,v;**

**u = stoi(rev(nums[i]));**

**v = stoi(rev(nums[i+1]));**

**if(u>v) swap(nums[i],nums[i+1]);**

**}**

**for(auto number: nums) cout<<number<<endl;**

**}**

#### **Stock Market**

**Note:** **Students using C, C++ or Java to solve the problem have to use long int for the variables instead of int. The results are different and using long int gives the correct answer. The problem description is also modifed to add more clarity of what has to be done to handle overflow. Please try again with these suggestions so that your code may pass now**

In the stock market of the city Knowhere, the stock prices of a given day is calculated as 3 times the price of previous day plus 4 times the price of the day before yesterday.  
Given the prices on day 1 and day 2, answer Q queries asking you about the price of the stock on the Nth day.

**Input Format:**  
The first line of the input contains two space-separated integers A and B, denoting the initial price of stock on day 1 and day 2  
The next line of the input contains an integer Q denoting the number of queries.  
The next line contains Q space-separated integers each denoting the number of the day for which the stock price is asked.

**Output Format:**  
Print one line containing Q space-separated integer, ith of them denoting the value of the stock price on ith day.**Since there is a possibility of overflow of values while calculating price on ith day (i.e.) values may become larger than the value contained in 32 bit integer, so take modulo (10^9+7) wherever required.**

**Constraints:**  
1 <= A,B <= 10^5  
1 <= Qi <= 10^3

**Example:**  
**Input:**  
1 2  
3  
1 2 3

**Output:**  
1 2 10

**ANS.**

**Brute force iterative:**

**a,b = map(int,input().split())**

**q,queries = int(input()),list(map(int,input().split()))**

**for i in queries:**

**if i == 1: print(a,end = " ")**

**elif i == 2: print(b,end = " ")**

**else:**

**ans = 0**

**day\_pre,day\_pre1 = b,a**

**for j in range(i-2):**

**ans = (day\_pre\*3+day\_pre1\*4)%1000000007**

**day\_pre1 = day\_pre**

**day\_pre = ans**

**print(ans,end = " ")**

**better solution using iterative dp:**

**a,b = map(int,input().split())**

**q,queries = int(input()),list(map(int,input().split()))**

**ans = [a,b]**

**for i in range(max(queries)-2):**

**ans.append((3\*ans[-1]+4\*ans[-2])%1000000007)**

**for i in queries: print(ans[i-1],end = " ")**

**only recursive solution (didn’t run due to time limit but is correct):**

**def solver(a,b,n):**

**if n == 1: return a**

**if n == 2: return b**

**return (solver(a,b,n-1)\*3+solver(a,b,n-2)\*4)%1000000007**

**a,b = map(int,input().split())**

**q,queries = int(input()),list(map(int,input().split()))**

**for query in queries:**

**print(solver(a,b,query),end = " ")**

**recursive solution using dp (optimized recursion using memoization):**

**ans = [-1 for i in range(1001)] #array to store answers**

**def solver(a,b,n):**

**if ans[n] != -1: return ans[n] #if value asked for in array then return value from array**

**else: #else add it in the array (keep the base cases also here)**

**if n == 1: ans[1] = a; return ans[1]**

**elif n == 2: ans[2] = b; return ans[2]**

**else:**

**ans[n] = (solver(a,b,n-1)\*3+solver(a,b,n-2)\*4)%1000000007**

**return ans[n]**

**a,b = map(int,input().split())**

**q,queries = int(input()),list(map(int,input().split()))**

**for query in queries:**

**print(solver(a,b,query),end = " ")**

**steps: 1. recursion/iterative brute force**

**2. memoization in recursion (if possibly recursive soln is present)**

**3. memoization in iterative soln (shorter code)**

#### **Row Column Pairing of matrices**

Given two matrices m1 , m2 and a capacity 'C', write a code to obtain a matrix m3 after row-column pairing. Row - column paring is an operation in which the elements in the matrix1 are paired rowwise with elements of matrix2 columnwise to obtain a matrix m3. m3[i][j] is the count of elements of ith row when added with corressponding (column index of element in matrix1 is same as row index of element in matrix2) elements of jth column is less than or equal to capacity C.

For example, given a matrix m1 as below:

120 230 170 110

100 90 115 190

135 78 95 245

and m2 as below:

300 400 240 450

456 230 245 350

236 230 340 234

392 321 295 331

when capacity is 500, matrix m3 will be:

2 3 3 2

2 2 4 2

2 2 3 2

**Input Format**

First line contains the number of rows in matrix1, r1

Next line contains the number of columns in matrix1, c1

Next r1 contains the elements in row1 of matrix1 separated by a space

Next line contains the number of rows in matrix2, r2

Next line contains the number of columns in matrix2, c2

Next r2 contains the elements in row1 of matrix2 separated by a space

Next line contains the capacity, s

**Output Format**

Print matrix m3 elements of one row in a single line separated by a tab

**Note:**A tab is present at the end of each row

CASES:

Input

3

4

120 230 170 110

100 90 115 190

135 78 95 245

4

4

300 400 240 450

456 230 245 350

236 230 340 234

392 321 295 331

500

Expected output

2 3 3 2

2 3 4 2

2 2 3 2

**ANS.**

**r1,c1 = int(input()),int(input())**

**m1 = [list(map(int,input().split())) for i in range(r1)]**

**r2,c2 = int(input()),int(input())**

**m2 = [list(map(int,input().split())) for i in range(r2)]**

**c = int(input())**

**#TRANSPOSE TO BRING COLUMNS INTO ROWS FOR EASY ACCESS**

**m2\_t = []**

**for i in range(c2):**

**row = []**

**for j in range(r2):**

**row.append(m2[j][i])**

**m2\_t.append(row)**

**m3 = [[0 for j in range(c2)] for i in range(r1)]**

**for i in range(r1):**

**for j in range(c2):**

**row,column = m1[i],m2\_t[j]**

**for k in range(c2):**

**if row[k]+column[k] <= c: m3[i][j]+=1**

**for i in range(r1):**

**for j in range(c2):**

**print(m3[i][j],end = "\t")**

**print()**

**Minimum Multiple**

Given a collection C1 of ‘n’ positive integers and a number ‘m’ write a C program to find the minimum multiple of m in C1. If no such multiple exist in C1 then print ‘No multiple found’

For example, if there are seven elements 23, 24, 25, 12, 6, 7, 11 and m is 3 then the output should be 6.

**Input Format**

First line contains the number of elements in the collection C1, n

Next ‘n’ lines contain the elements in the collection C1

Next line contains the value of m

**Output Format**

Print the minimum multiple of ‘m’ present in C1 or ‘No multiple found’

**ANS.**

**l=sorted([int(input()) for \_ in range(int(input()))])**

**m=int(input())**

**import sys**

**for i in l:**

**if i%m==0:**

**print(i)**

**sys.exit()**

**print("No multiple found")**

**better solution**

**nums = [int(input()) for i in range(int(input()))]**

**m,min = int(input()),10\*\*9**

**for i in nums:**

**if i%m == 0 and i < min: min = i**

**if min == 10\*\*9: print("No multiple found")**

**else: print(min)**

**Substrings without Vowels**

A substring is a contiguous sequence of characters within a string. Given a string S and an integer n, write a C code to generate all substrings of length n without vowels. For example, if the given string S1 is ‘apple’ then substrings of S1 of length 3 are app, ppl, pple. And substrings of length 3 without vowels is only one and it is ppl.

**Note:**

In C language when a string is prepared by programmer by putting characters in a character array, he has to place a ‘\0’ (null character) at the end explicitly

**Boundary Conditions**

length of the string s <=20 without spaces

n>0 and n<length of s

There is atleast one substring of length n without vowels

**Input Format**

First line contains a string, S in lower case

Next line contains an integer, n

**Output Format**

Print substrings of length n without vowels

One substring in one line

**ANS.**

**s=input()**

**n=int(input())**

**l=[]**

**for i in range(len(s)):**

**subStr = ""**

**for j in range(i,len(s)):**

**subStr += s[j];**

**if len(subStr)==n:**

**l.append(subStr)**

**l1=[]**

**vowels=['a','e','i','o','u']**

**for i in l:**

**for letters in i:**

**if letters in vowels:**

**l1.append(i)**

**for j in l:**

**if j not in l1:**

**print(j)**

**better approach:**

**def check(s):**

**for i in s:**

**if i in "aeiou":**

**return 1**

**return 0**

**s,n = input(),int(input())**

**for i in range(len(s)):**

**for j in range(i+1,len(s)+1):**

**if len(s[i:j]) == n and check(s[i:j]) == 0:**

**print(s[i:j])**

**best approach:**

**s,n = input(),int(input())**

**for i in range(len(s)+1-n):**

**sub = s[i:i+n]**

**if not('a' in sub or 'e' in sub or 'i' in sub or 'o' in sub or 'u' in sub):**

**print(sub)**

**C CODE:**

**#include <stdio.h>**

**#include <string.h>**

**void solver(char s[],int begin,int end){**

**char check[end-begin];**

**int j = 0,flag = 0;**

**for(int i=begin;i<end;i++){**

**if(s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u'){**

**flag = 1;**

**break;**

**}**

**check[j++] = s[i];**

**}**

**if(flag == 0)**

**printf("%s\n",check);**

**}**

**int main(){**

**char string[20];**

**scanf("%s",string);**

**int n;**

**scanf("%d",&n);**

**for(int i=0;i<strlen(string)+1-n;i++) solver(string,i,i+n);**

**}**

**Find the Sum of Error**

A communication company transmits two series of integers S1 and S2 from one city to the other. During the transmission the series S2 got corrupted. The company deleted the orginal data after transmission but they have the XOR (S1,S2) which contains the XOR values of the corressponding eleements of S1 and S2. Interesting fact about XOR operation is if a XOR b = c then a XOR c = b and b XOR c = a. Given the number of integers ‘n’ in the series S1 and S2, elements of the series S1 and S2 at the recieving end, XOR(S1, S2), write a C program to find the original elements of s2 and the sum of errors in the transmission. For example, if five integers were transmitted and the series S1 and S2 at the receiving end are

S1 - 17, 29, 35, 48, 93

S2 – 45, 67, 108, 34, 56

and the XOR (S1, S2) is

59, 96, 75, 23, 103

Then the orginal S2 series is

42, 125, 104, 39, 58

and the sum of error is 72

**Input Format**

First line contains the number of elements in the series, n

Next ‘n’ lines contain the elements of series, s1

Next ‘n’ lines contain the elements of series, s2

Next ‘n’ lines contain the elements of series, XOR(s1,s2)

**Ouput Format**

Print the orignial elements of the series, s2 in n lines

Print the sum of error

**Note:**

Absolute value of a number x can be found in GCC by using abs(x) which is declared in the header file stdlib.h

**ANS.**

**n = int(input())**

**s1 = [int(input()) for i in range(n)]**

**s2\_inc = [int(input()) for i in range(n)]**

**xor = [int(input()) for i in range(n)]**

**sum\_of\_errs = 0**

**for i in range(n):**

**x = s1[i]^xor[i]**

**sum\_of\_errs += abs(x-s2\_inc[i])**

**print(x)**

**print(sum\_of\_errs)**

**Common Divisor**

You are given a sequence of integers. You want to find out if all the numbers have a common divisor. Common divisor should be greater than 1.  
If there exists such a number, print "YES", otherwise print "NO".

**Input Format:**The first line of the input contains an integer N, denoting the number of elements in the array.  
The second line contains N space-separated integers.

**Output Format:**Print "YES" or "NO" on a single line(without quotes) according to the above condition.

**Example:  
Input:**3  
2 3 4  
**Output:**NO

**Input:**5  
5 10 15 20 25

**Output:**YES

ANS.

**from fractions import gcd**

**from functools import reduce**

**def find\_gcd(list):**

**x = reduce(gcd,list)**

**return x**

**n=int(input())**

**l=list(map(int,input().split()))**

**a=find\_gcd(l)**

**if a>1:**

**print("YES")**

**else:**

**print("NO")**

**C++**

**#include <iostream>**

**using namespace std;**

**int gcd(int a,int b){**

**if(b == 0) return a;**

**else return gcd(b,a%b);**

**}**

**int gcd\_arr(int arr[],int n){**

**int ans = arr[0];**

**for(int i=1;i<n;i++){**

**ans = gcd(ans,arr[i]);**

**}**

**return ans;**

**}**

**int main(){**

**int n;**

**cin>>n;**

**int arr[n];**

**for(int i=0;i<n;i++) cin>>arr[i];**

**if(gcd\_arr(arr,n) == 1) cout<<"NO";**

**else cout<<"YES";**

**}**

**Stepped Strings**

Two equal length strings S1 and S2 are said to be stepped strings if the difference between the letter position in English alphabet of non space characters is ‘i’ where ‘i’ is the position of the character in the strings excluding spaces in the strings. Letter position of character ‘a’ in English alphabet is 1, letter position of character ‘b’ is 2 and so on.

Given two strings S1 and S2, write a C program to check if S1 and S2 are stepped strings. If the strings are of different length then print  "Length different" and if strings contain non alphabets then print "Strings contain non alphabets" and if strings differ in the position of space then print  "Strings differ in space" and print  "Strings are not stepped" when valid strings are given and they differ in the step value

**Input Format:**

First line contains the string, s

**Output Format:** Print either Strings are stepped or Length different or Strings differ in space or Strings are not stepped or Strings contain non alphabets

**ANS.**

**alp = "abcdefghijklmnopqrstuvwxyz" # I could have used a dict here to bind positions with letters**

**a,b,space1,space2 = list(input()),list(input()),0,0**

**if len(a) != len(b): print("Length different"); exit()**

**for i in a: # I could have used index-based for loop for more efficiency**

**if i == ' ': space1 = a.index(i); continue**

**if not(97<=ord(i)<=122):**

**print("Strings contain non alphabets"); exit()**

**for i in b:**

**if i == ' ': space2 = b.index(i); continue**

**if not(97<=ord(i)<=122):**

**print("Strings contain non alphabets"); exit()**

**if space1 != space2: print("Strings differ in space"); exit()**

**a.remove(' '); b.remove(' ')**

**for i in range(len(a)):**

**if(abs(alp.index(a[i])-alp.index(b[i])) != i+1):**

**print("Strings are not stepped"); exit()**

**print("Strings are stepped")**

**Rectangle equivalent to Square**

Given  the area (an integer) of a square (a closed polygon with four sides whose lengths are same), write an algorithm and the code to compute the number of rectangles ( a closed polygon with fours sides, whose opposite sides are of same length and  lenght of the adjacent sides are not same) that can be formed whose area is equal to the area of the given square. The dimensions of a rectangle are called as length of the rectangle(an integer) and the breadth of the rectangle (an integer), represented by 'lengthX breadth'.  The dimension of a rectangle with lenghth 4 units and breadth 2 units is written as 4X2. Area of a rectangle is the product of the length and the breadth of the rectangle.  For exampe, if the area of the square is 36 sqaure units, the possible rectangles that can be formed whose area is 36 square units are:  rectangles with dimension : 36X1, 18X2, 12X3, 9X4.  The rectangles with the dimension 18X2 and the rectangle with the dimension 2X18 are considered as the same rectangle. Hence, for the given square with area 36 square units, four rectangles can be formed.

**Input format:**

Enter the area of the square

**Output format:**

Number of rectangles that can be formed, whose area is equal to the given area.

**ANS.**

**from math import sqrt**

**area,cnt = int(input()),0**

**for i in range(1,int(sqrt(area))+1):**

**if area%i: pass**

**else:**

**if area//i != i: cnt+=1**

**print(cnt)**

**Non-triangle triplet**

In a triangle, sum of the lengths of two sides should be greater than the length of the third side.  A triplet of integers is a collection of three integers. For eg., (1,2,3) is a triplet of integers. The triplets (1,2,3)   (2,3,1) , (3,1,2), are considered to be the same triplets. Given a positive integer N, Write a code to compute all the distinct triplets (a,b,c) such that a+b+c = N and the triplet (a,b,c) does not form the length of sides of  a triangle. If N=5, of the two possible triplets: (1,1,3), (1,2,2), the triplet (1,1,3) does not form the sides of a triangle.

**Triangle Rule :** lengths of any two sides of a triangle must add up to more than the length of the third side.

**Input format:**

Enter the number N

**Output Format :**

Number of  distinct triplets (a,b,c) such that a+b+c=N and the triplet (a,b,c) does not form the sides of a triangle.

**ANS.**

**Monish’s code:**

**from math import ceil**

**n = int(input())**

**triplets = []**

**for c in range(ceil(n/2), n-1):**

**for a in range(1, n-c):**

**b = n-a-c**

**if (a, b, c) in triplets or (b, a, c) in triplets:**

**continue**

**triplets.append((a, b, c))**

**print(len(triplets))**

**easy and efficient solution:**

**n = int(input())**

**count = 0**

**vis = set()**

**for i in range(1,n//2):**

**for j in range(1,n//2):**

**k = n-i-j**

**x = tuple(sorted([i,j,k]))**

**if not(i+j>k and i+k>j and j+k>i) and x not in vis:**

**count+=1; vis.add(x)**

**print(count)**

**Star reducible word**

A word W (sequence of symbols from A-Z, a-z ) is reduced to another word by  the application of the following rules.

Rule 1: If the word contains a sequence of same symbols (that is, same symbol occurs more than once, in the word) that sequence is replaced by \*. For example, if the word is abbcde,  then by rule1, we have a \*cde. Here, bb is replaced by \*.

Rule 2). If we have \* before any symbol, say \*a, we replace \*a by a.

Similarly, if we have \*, after any symbol, say, a \*, we replace 'a \* ' by a.

Rule 3). If we have more than one , successive stars, we can replace that successive stars by just one \*. For eg, the word 'a \* \*bc' gets reduced to 'a\*bc'.  Note that, by rule 2, a\*bc will get reduced to abc.

For example, the word

'cabbabbbc' gets

reduced to \* as follows.

Step 1: by rule1, the word 'cabbabbbc' gets reduced to ca\*abbbc

Step 2: by rule2, the word ca\*abbbc gets reduced to caabbbc

Step 3: by rule 1, the word caabbbc gets reduced to c\*bbbc

Step 4: by rule 2, the word c\*bbbc gets reduced to cbbbc

Step 5: by rule 1, cbbbc gets reduced to c\*c

Step 6: by rule 2, c\*c gets reduced to cc

Step 7. By rule 1, cc gets reduced to \*.

Thus, the word cabbabbbc gets reduced to \*.

You can apply the rules in any order, any number of times.

The word abbcaa will get reduced to ac. We cannot reduce this word further.

Given a word w, write a code to check whether the given word is reducible to \* or not. If the word is reducible to \*, your code should out put 1. If the given word is not reducible to \* by the application of any rule, any number of times, your code should out put 0.

All the words are case insensitive.

**Input format**

Enter the word

**Output**

0 or 1 depending on the reducibility.

**ANS.**

**c=input()**

**s=c.lower()**

**snew1=""**

**snew2=""**

**for i in range(len(s)):**

**for j in range(i+1,len(s)+1):**

**if len(s[i:j])>1 and len(set(list(s[i:j])))==1:**

**snew=s.replace(s[i:j],"\*")**

**for l in range (len(snew)):**

**if snew[l]=="\*":**

**snew1=snew.replace(s[l-1:l],"\*")**

**if snew1.count("\*") > 1:**

**a="\*"\*(snew1.count("\*"))**

**snew2=snew1.replace(a,"\*")**

**print(len(snew2))**

**BETTER APPROACH:**

**w = input()**

**if len(set(list(w))) == len(w) and 'a' not in w: print(0)**

**else: print(1)**

#### **Matching index of a word and a number**

With the help of the following procedure, every word is represented as an integer with even number of digits and an integer with even number of digits,  is represented as a word,

Every symbol, A to Z is represented by the integers from 01 to 26 and vice versa. . The symbols from 'a' to 'z' is represented by the integers , from 27 to 52  A represents 01, B represents 02, and so on. Z represents 26.  Similarly  'a' represents 27, 'b' represents 28 and so on. z represents 52. The word  `abc' is represented by the intger 272829. Every symbol in `abc' is replaced by the respective integers.  `AbC' is represented by the integer 012803.  Note that the integer 012803 is not the same as the integer 12803.

 By the following procedure, any  positive integer with even number of digits is represented as  a word ,

Every two successive digits, from the left (or right)  are replaced by the respective  symbols.   27 is replaced by 'a', 01 is replaced by 'A', 26 is replaced by 'Z'.  If the successive digits, treated as a 2-digit integer , is greater than 52, that 2-digit intger is replaced by the word  that represents the number that equals 2-digit integer mod 52.

57 mod 52 =05.  Hence, 57 is replaced by the word that corresponds to 05.  57 is replaced by E.

By the above procedures, every word is represented as a positive integer with even number of digits and vice versa.

234501 is represented by WsA.  043131422749273835 represents the word 'Deepawali'.  Diwali is represented  by the integer 043549273835.

A word  'W' and an integer 'N'  (with even number of digits)  is said to be a perfect match if the intger representation of the word 'W' is equal to the integer N. In that case, the first two digits (from the left) correspond to the first symbold of the word.  The third and fourth digits, treated as a 2-digit integer corresponds to second symbol of the word and so on.  If 'W' and 'N' are a perfect match, all the succesive 2-digits of N is mapped  correctly, to the successive symbols of W.  In that case, the matching index of W and N is equal to the length of the word.

The integer 234501 perfectly matches with the word  WsA  and the matching index of 234501 & WsA is 3.  All the three successive 2-digits of the integer  maps correctly to the three symbols of WsA.

The integer 234501 does not perfectly match with  WsB , since the digit 01 does not match correclty  to the symbol 'B' and the other symbols match correctly with the respective 2-digits.  In that case , matching index of 234501 & WsB is 2. Similarly,  The matching index of 234501 & WtB is one, since the symbols 't' and 'B' do not match correctly.

The matching index of 234501 & XtB is 0, since none of the symbols of the word do not match correctly with the respective integers.

Given a word W & an integer N (with even number of digits), matching index of W & N is the number of symbols of W that match correctly with the respective 2-digits of N .

Given a word W and an integer N(with even number of digits), write a code to compute the matching index of W & N.  If the number of digits in N is not equal to twice the number of symbols in W,    your program should output -1.

**Input format:**

Enter the integer N

Enter the word W

**Output format**

Print the matching index of N & W

**ANS.**

**n=input()**

**w=input()**

**if len(n)==2\*len(w):**

**l=[]**

**for i in range (0,len(n),2):**

**l.append(n[i:i+2])**

**alpha1=[]**

**alpha2=[]**

**for i in range(97,123):**

**alpha1.append(chr(i))**

**for j in range (65,91):**

**alpha2.append(chr(j))**

**alpha=alpha2+alpha1**

**num=['01','02','03','04','05','06','07','08','09']**

**for i in range(10,53):**

**num.append(str(i))**

**{**

**ALTERNATE WAY:**

**for i in range(1,53):**

**if i<10:**

**num.append(‘0’+str(i))**

**else:**

**num.append(str(i))**

**}**

**info=dict(zip(num,alpha))**

**l1=[]**

**for i in l:**

**d=int(i)**

**if d<53:**

**l1.append(info[i])**

**else:**

**a=52%(int(i))**

**if a>9:**

**l1.append(info[str(a)])**

**else:**

**b='0'+str(a)**

**l1.append(info[b])**

**c="".join(l1)**

**count=0**

**for i in c:**

**if i in w:**

**count+=1**

**print(count)**

**else:**

**print(-1)**

**BETTER SOLUTION CODE:**

**n,w,cnt = input(),input(),0**

**if len(n)%2 != 0: print(-1); exit()**

**alp = {}**

**start\_l = ord('a')-1**

**start\_u = ord('A')-1**

**for i in range(1,27):**

**if i<10: alp['0'+str(i)],alp[str(26+i)] = chr(start\_u+i),chr(start\_l+i)**

**else: alp[str(i)],alp[str(26+i)] = chr(start\_u+i),chr(start\_l+i)**

**for i in range(0,len(n),2):**

**x,y = n[i:i+2],w[i//2]**

**if(alp[x] == y): cnt+=1**

**print(cnt)**

#### **Pair with same divisors**

An integer n is said to be the divisor of another integer m if m leaves a remainder 0 when divided by n. We define a function : d(N) which is the number of pairs  (a,b) such that, a<b,  a\*b=N and both the integers a and b have the same number of divisors.     For 24, we can have the pairs : (2,12), (3,8),(4,6), (1,24),(8,3).  Here in any of the pair, the number of divisors of both the integers in a pair is not the  same. Hence, d(24)=0. For 48, the pairs are : (3,16),(2,24),  (4,12) , (1,48)and (6,8). Here the pair (6,8) is a pair such that both 6 and 8 have 4 divisors.  That is, (6,8) is a pair in which both the integers have same number of divisors. Other pairs of 48, do not have this property. Hence d(48)=1.

Given a positive integer N, Write a code to compute the value of d(N).  For a given number N, if no pairs  (a,b) such that a<b, a\*b=N, are possible, your code should output -1

**Input format:**

Enter the number N

**Output format :**

Value of d(N)

**ANS:**

**def divisor(n):**

**count1=0**

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

**if n%i==0:**

**count1+=1**

**return count1**

**n=int(input())**

**l=[]**

**for i in range(1,n//2+1):**

**if n%i==0 and (n//i)>i:**

**l.append([i,n//i])**

**count=0**

**for i in l:**

**if divisor(i[0])==divisor(i[1]):**

**count+=1**

**if len(l)==0:**

**print(-1)**

**else:**

**print(count)**

**BETTER SOLUTION:**

**from math import sqrt**

**def no\_of\_div(n):**

**count = 0**

**for i in range(1,int(sqrt(n))+1):**

**if n%i == 0:**

**if n//i != i: count+=2**

**else: count+=1**

**return count**

**n,ans,flag = int(input()),0,0**

**for i in range(1,int(sqrt(n))+1):**

**if n%i == 0:**

**if i != 1: flag = 1**

**if n//i != i:**

**if no\_of\_div(n//i) == no\_of\_div(i): ans+=1**

**if flag == 0: print(-1)**

**else: print(ans)**

#### **Transformation the word**

We introduce a  function , denoted by P such that P : {1,2,3...n} -----> {1,2,3,...n}.  Foe eg, if  P(1)=2, P(2)=3, ... P(n-1)=n, p(n)=1; then P is represented by a sequence : 2,3,4.,..1.   2 that appears in the first position indicates that, the image of 1 is 2,  1 appears in the last position informs that the image of n is 1, and so on.  For a set with n elements, n! functions are possible.   .  We can apply this function P on a word , to transform the word.

Given a word abcd, and a function P : 2,3,4,1 The function tranforms the word as follows:

Since the image of 1 is 2, the symbols in the first position moves to  the second position. Since the image  of 2 is 3.  the symbol in the second position moves to the third position. Since the image of 4 is 1, the symbol in the fourth position moves to the first position.  Hence P(abcd) = dabc, where P = 2,3,4,1.

If P= 2,1,3 , P(abd) = bad. The symbol in the first position moves to the second position, the symbol in the second position moves to the first position and the symbol in the third position remains in the third position.  A function P that acts upon a word of length n, should have a sequence of n numbers, separated by a comma. If we apply the function: 2,1,3 to bad, we get the original word abd.

Given a word w and a function  P,  Write a code to compute P(w) and to compute the function which will transform P(w) to w.

**Input format:**

Enter the word of length n

Enter the first digit of the function

Enter the second digit of the function.

...

Enter the n-th digit of the function

**Output format :**

Print the word P(w)

The next n-lines should print the digits of the function that transforms P(w) to w.

Illustartion

Input :

abc

2

3

1

Output:

cab

3

1

2

**ANS:**

**s=input()**

**w=list(s)**

**l=[int(input()) for \_ in range(len(s))]**

**info=dict(zip(l,w))**

**l1=[info[i+1] for i in range(len(w))]**

**print("".join(l1))**

**index=[w.index(j)+1 for j in w]**

**info1=dict(zip(w,index))**

**for k in l1:**

**print(info1[k])**

**SHREYASH’S CODE:**

**st=input()**

**lt=list(st)**

**flt=lt.copy()**

**for i in st:**

**flt[int(input())-1]=i**

**final = "".join(flt)**

**print(final)**

**for i in final:**

**print(lt.index(i)+1)**

**MY BETTER SOLUTION:**

**s = input()**

**ans,k,info = ['0']\*len(s),0,{}**

**for i in range(len(s)):**

**m = int(input())**

**ans[m-1] = s[k]**

**k+=1**

**info[ans[m-1]] = k**

**print("".join(ans))**

**for letter in ans: print(info[letter])**

**Q. Given a matrix, with the given matrix create a new matrix by converting the row's in matrix to column's and column's as row's. Add both the matrix and print the result. Write this python program using functions by passing arguments. create separate function for input, transpose, addition and output.**

**ANS.**

def inpt():  
 order=int(input())  
 matrix=[[int(input()) for \_ in range(order)] for \_ in range(order)]  
 return matrix  
  
def transpose(omatrix):  
 matrix=[i.copy() for i in omatrix]  
 for i in range(len(matrix)):  
 for j in range(i+1,len(matrix)):  
 matrix[i][j],matrix[j][i]=matrix[j][i],matrix[i][j]  
 return matrix

def addition(original,transmatrix):  
 msum=[[i+j for i,j in zip(a,b)] for a,b in zip(original,transmatrix)]  
 return msum  
  
def printmatrix(matrix):  
 for i in matrix:  
 print(i)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 original=inpt()  
 transmatrix=transpose(original)  
 finalmatrix=addition(original,transmatrix)  
 printmatrix(finalmatrix)

**ALTERNATE CODE:**

#without functions  
order=int(input())  
matrix=[[int(input()) for \_ in range(order)] for \_ in range(order)]  
orig=[i.copy() for i in matrix]  
for i in range(order):  
 for j in range(i+1,order):  
 matrix[i][j],matrix[j][i]=matrix[j][i],matrix[i][j]  
msum=[[i+j for i,j in zip(a,b)] for a,b in zip(orig,matrix)]  
for i in msum:  
 print(i)

#### **Difference of the sum of alternate digits (D)**

Given a positive integer 'x' (with even number of digits in it), write an algorithm and the subsequent code to compute the difference between  the sum of the digits occuring in the alternate positions (starting from the first position) and the sum of the digits occuring in the alternate positions,starting from the last rightmost position of 'x'

For example, consider the number  8975.  The sum of the digits that occur in the alternate positions from the first position is 8+7=15.  The sum of the digits that occur in the alternate positions, starting from the rightmost position is 5+9 = 14. Difference between the two sums is 1 (=15-14).  Similarly, for the number 5798, the difference between  two sums, is 1.

**Note: Read the input as a number and do entire processing as  a number**

**C++ compilers can compile C code also**

**Input format**

First line contains the positive integer

**Output format :**

First line should contain the difference between the sum of the digits occurring in the alternate positions (starting from the first position) and the sum of the digits occurring in the alternate positions (starting from the last rightmost position).

**ANS.**

**n=int(input())**

**s=str(n)**

**s1=0**

**s2=0**

**for i in range(0,len(s),2):**

**s1+=int(s[i])**

**for i in range (1,len(s),2):**

**s2+=int(s[i])**

**print(abs(s1-s2))**

#### **Substrings and Sweets**

Ramu, a small boy loves to eat sweets. Whenever he gets time, he goes to a local sweet shop and buys sweets, but his father is really worried about Ramu’s Health and wants to make sure that he doesn’t eat more sweets. So he indulged Ramu in an interesting game. Everyday he will give Ramu a word 'w' and a number 'k'. Ramu will be allowed to buy only the number of sweets equal to the number of substrings with 'k' vowels that he generates from the given word 'w' . Ramu wants to get the maximum number of possible sweets. But Ramu doesn’t know what a substring is and also do not know how to count numbers, fortunately, he knows that you are a good programmer, so he has given this task to you. Given the word 'w' and number 'k', write a code to print the number of sweets that Ramu can get.

**Vowels: a,e,i,o,u**

A **substring** is a contiguous sequence of characters within a string. For example, string apple all possible substrings of length greater than or equal to one are apple, appl, pple, app, ppl, ple, ap, pp, pl, le, a, p, p, l, e.

**Input Format:**

There are two lines of input,

The first line contains the word given by Ramu’s father and,

The second line contains the value of k.

**Output Format:**

Print only one integer, containing the value of sweets that Ramu can buy.

**Example:**

abc

1

All substrings of abc are:

 a, ab, abc, b, bc, c

Out of all these substrings a, ab and abc have 1 vowel exactly.

Hence the answer is 3.

**ANS.**

**p=input()**

**w=p.lower()**

**k=int(input())**

**substrings=[]**

**for i in range(len(w)+1):**

**for j in range(i+1,len(w)+1):**

**substrings.append(w[i:j])**

**vowels=['a','e','i','o','u']**

**count1=0**

**for i in substrings:**

**count=0**

**for j in i:**

**if j in vowels:**

**count+=1**

**if count==k:**

**count1+=1**

**print(count1)**

**Togetherness of a Family**

**In the world of magic land, there is a park in which a group of people are playing a game. Every person in the group is a child of one another person in the group except one person who is the parent of all the people in the group. Here, each person is represented by a number that if there are ‘n’ persons in the group then they are represented by numbers 1 to 7. The direct and indirect child relationship is defined as follows:**

1. **A person ‘b’ is said to be direct child of ‘a’ if ‘a’ is parent of ‘b’.**
2. **Person is ‘c’ is said to be indirect child of ‘a’ when ‘c’ is child of ‘b’ and ‘b’ is child of ‘a’.**
3. **A indirect child ‘i’ of a person ‘p’ is also an indirect child of parent of ‘p’.**

**In the game, everyone is given a token which has a number written on it and to test the togetherness of the family, the family is given some queries.**

**In one query, a number representing a person is given and it is asked to find the number of direct and indirect children of the person who have prime number in the token given to them.**

**For more clarifications see the example below.**

**There are seven person in the group and the tokens assigned to each of them are:**

**Person**

**Token Number**

**1**

**1**

**2**

**2**

**3**

**3**

**4**

**4**

**5**

**5**

**6**

**6**

**7**

**7**

**We are given 2 queries:**

**Query1: The person numbered as 1 is given and we need to findout the number of children(both direct and indirect under 1) which have a token with prime number value:**

**Answer: 4.**

**Explanation:**

**For the person numbered as 1, its direct children are persons numbered as 2 and 3 and they have tokens 2 and 3 respectively which are prime numbers.**

**The person numbered as 5 and 7 are also children (indirect) of the person number 1 hence they also added up to the count and hence there are a total of 4 people who are children of person number 1 and have a prime number token.**

**Query2: The person numbered as 6 is given**

**Answer: 0.**

**Explanation:**

**The person with number 6 has no children and hence there are 0 children under him having tokens with prime number values.**

**Input Format:**

**First line of the input contains N, the number of persons playing in the park**

**Second line contains token numbers assigned for the persons and they are separated by a space ith number in the line contains the value of the token assigned to the ith person.**

**Next N-1 lines contains two integers Xi and Yi separated by a space and indicates that Xi is the parent of Yi.**

**The next line contains the number of queries, Q**

**Next Q lines contain the person queried**

**Note: The persons are indexed from 1.**

**Output Format:**

**Output Q lines, each containing only 1 integer, the number of children for the given node, which have their token value as a prime number**

**Example:**

**Input:**

**7**

**1 2 3 4 5 6 7**

**1 2**

**1 3**

**2 4**

**2 5**

**3 6**

**3 7**

**2**

**1**

**6**

**Output:**

**4**

**0**

**As can be seen**

**1 is the parent of both 2 and 3.**

**2 is the parent of 4 and 5.**

**3 is the parent of 6 and 7**

**Test case:**

7

1 2 3 4 5 6 7

1 2

1 3

2 4

2 5

3 6

3 7

2

1

6

Expected output

4

0

**PYTHON CODE:**

import math

# Number of people

n = *int*(*input*())

tokens = [] # For storing tokens

tree = [] # For storing adjacency list in tree, check https://youtu.be/09zltCNaRF8 for more

result = [] # For storing the results

#Slightly improved logic to check if a number is prime or not

def isPrime(number):

if(number <= 1): return False

if(number <= 3): return True

if(number % 2 == 0 or number % 3 == 0):

return False

for i in *range*(5, math.ceil(math.sqrt(number)) + 1, 6):

if(number % i == 0 or (number % (i + 2) == 0)):

return False

return True

# Depth first traversal as explained:https://youtu.be/pEwTw97BS2c?t=340

def depth\_first\_traversal(source):

for child in tree[source]:

#Go deep into the branch

depth\_first\_traversal(child)

#While coming up, bubble up the results from children to the parent

for child in tree[source]:

if(isPrime(tokens[child-1])):

result[source] += 1

result[source] += result[child]

#Get all the tokens

tokens = *list*(*map*(*int*, *input*().split()))

#Initialise tree and results

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

tree.append([])

result.append(0)

#Store the tree as adjacency list

for i in *range*(n-1):

x,y = *map*(*int*, *input*().split())

tree[x].append(y)

#Perform depth first traversal and build the results list,

#this operation only needs to run once

depth\_first\_traversal(1)

#Answer all the queries

q = *int*(*input*())

for \_ in *range*(q):

x = *int*(*input*())

print(result[x])

**C++ CODE:**

#include <iostream>

#include <vector>

using namespace std;

const int MAXN = 1e3+5;

int result[MAXN];

bool isPrime(int number) {

if(number <= 1) return *false*;

if(number <= 3) return *true*;

if(number % 2 == 0 || number % 3 == 0) {

return *false*;

}

for(int i = 5; i\*i <= number; i+= 6) {

if(number % i == 0 || number % (i+2) == 0) {

return *false*;

}

}

return *true*;

}

void depth\_first\_traversal(int source, vector<vector<int>>& tree, vector<int> tokens) {

for(int *child*: tree[source]) {

depth\_first\_traversal(child, tree, tokens);

}

for(int *child*: tree[source]) {

if(isPrime(tokens[child-1])) {

result[source] += 1;

}

result[source] += result[child];

}

}

int main() {

int n;

cin>>n;

int token;

vector<int> tokens;

vector<vector<int>> tree(n+1, vector<int>());

for(int i = 0; i < n; i++) {

cin>>token;

tokens.push\_back(token);

}

int x, y;

for(int i = 0; i < n-1; i++) {

cin>>x>>y;

tree[x].push\_back(y);

}

depth\_first\_traversal(1, tree, tokens);

int q;

cin>>q;

while(q--) {

cin>>x;

cout<<result[x]<<endl;

}

return 0;

}

Selection of lucky winner

In a cinema theatre named 'w', with `n' seats, whenever the theatre is full (i.e., when all the seats are occupied), theatre administartion will be gifting a viewer with a surprise gift and the lucky viewer will be chosen through the following process, called as the  'Character allottment process'.

**Character allotment process :**

1. Starting from the first seat, allot a character  of `w' ( starting from the first character) to each seat.  That is, allot the first character of w to the seat no.1, allot the second character of  `w' to the seat no.2 and so on.

2. If all the characters of `w' are allotted, then continue the allotment from the first symbol of `w' again.

3. After allotting a character to the last seat, continue the allotment process by allotting the next character (character,  next to the character allotted to the last seat earlier) from the last seat

4. Continue allotting the characters  till the first seat.

5. In the process, all the seats would have two characters allotted to each seat, one character during the forward allottment process  (when the characters are allotted from the first seat to the last seat) and another during the backward allottement process. (when the characters are allotted from the last seat to the first seat)

6. The first seat which gets the same character during both the forward allottment process and the backward allottment process is the lucky seat.

7. The person occupying the luck seat is given the surprise gift.

For eg, Name of the theatre : good.  Total number of seats : 10  During the forward allottment, seats are allotted as : 1-g, 2-o, 3-o, 4-d,5-g, 6-o, 7-o, 8-d, 9-g, 10-o

and during reverse allotment process 10-o, 9-d, 8-g, 7-o, 6-o, 5-d, 4-g, 3-o, 2-o, 1-d. The first seat to get the same characters in both the allottment is the seat no : 10.  The person who occupied the seat no. 10 is the lucky winner.

Given the name of the theatre `w', the total number of seats in the theatre, write an algorithm and the code to identify the lucky seat. Print -1 if no seat number could get the same letters.

**Input format :**

Name of the theatre: w

Total number of  seats in the theatre: n

**Output format :**

The seat number of the lcky seat.

**ANS.**

**import sys**

**w=input()**

**n=int(input())**

**forward={}**

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

**if i<=len(w):**

**forward[i]=w[i-1]**

**else:**

**forward[i]=w[(i%len(w))-1]**

**z=""**

**k=0**

**while((n%len(w))+k<len(w)):**

**z+=w[(n%len(w))+k]**

**k+=1**

**n1=(n-k)//len(w)**

**for i in range(n1+1):**

**z+=w**

**l=[i for i in range(n,0,-1)]**

**w1=list(z)**

**backward=dict(zip(l,w1))**

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

**if backward[i]==forward[i]:**

**print(i)**

**sys.exit()**

**print(-1)**

#### **Time in Heaven**

In the usual time-system, we have two time-formats: 24-hour format and the 12-hour format. 08:00:00 (hours:minutes:seconds) in the 24-hour format is written as 8:00:00 A.M in the 12-hour format. 17:25:32 in the 24-hour format is written as 05:25:32 P.M. 00:00:00 in the 24-hour format is written as 12:00:00 midnight. 12:00:00 in the 24-hour format is referred as 12:00:00 noon.

In a place called `Heaven’, the people follow an eight-hour format.

(i) 00:00:00 in the 24-hour format is written as 08:00:00 **midnight**.

(ii) Time in the range, from 00:00:01 hours to 07:59:59 in the 24-hour format is written as the same time with a suffix **A.M.** For eg: 07:00:10 in the 24-hour format is written as 07:00:10 A.M. in the 8-hour format.

(iii) 08:00:00 in the 24-hour format is written as 08:00:00 **midmorning**.

(iv) Time in the range, from 08:00:01 to 15:59:59, in the 24-hour format is written as the time in the range, from 00:00:01 to 07:59:59 , with a suffix **B.M.** For eg., 09:11:13 in the 24-hour format is written as 01:11:13 B.M .

(v) 16:00:00 in the 24-hour format is written as 08:00:00 **midevening**.

(vi)Time in the range, from 16:00:01 to 23:59:59 in the 24-hour format is written as the time in the range from 00:00:01 to 07:59:59, with a suffix **C.M.**For e.g., 17:59:59 in 24-hour format is written as 01:59:59 C.M

Given the time in the 12-hour format, Write an algorithm and the subsequent code to convert the given time into the time in `Heaven’ (ie., in to an 8-hour format)

**Input format :**

Time in the 12-hour format, hh:mm:ss followed by A.M or P.M or midnight

**Output format :**

Time in the 8-hour format, hh:mm:ss followed by A.M or B.M or C.M or midnight or midmorning

**Illustration :**

**Input**

12:00:00 midnight

**Output**

08:00:00 midnight

**Input**

02:00:11 P.M

**Output:**

06:00:11 B.M

**ANS: private test cases failed**

**import sys**

**usual = input()**

**heaven = ""**

**k = int(usual[0:2])**

**if usual == "00:00:00" or usual=="12:00:00 midnight" or usual=="12:00:00 A.M":**

**print("08:00:00 midnight")**

**sys.exit()**

**elif usual == "08:00:00" or usual=="08:00:00 A.M":**

**print("08:00:00 midmorning")**

**sys.exit()**

**elif usual == "16:00:00" or usual=="04:00:00 P.M":**

**print("08:00:00 midevening")**

**sys.exit()**

**elif usual=="12:00:00 noon" or usual=="12:00:00 P.M":**

**print("04:00:00 B.M")**

**sys.exit()**

**elif(k<=7 and k>=0):**

**heaven = usual**

**elif(k>=8 and k<=15):**

**k = k-8**

**b = "0" + str(k)**

**heaven = b + usual[2:8] +" "+"B.M"**

**elif(k>=16 and k<=23):**

**k = k-16**

**b = "0" + str(k)**

**heaven = b + usual[2:8] +" "+"C.M"**

**print(heaven)**

#### **Reverse Length Divisible**

Given a number **N,** check if it is reverse length divisible. A number is said to be reverse length divisible if the first i digits of the number is divisible by (l-i-1) where l is the number of digits in **N** and **0 < i <= l.**

For example, 52267 is reverse length divisible because:

5 is divisible by 5

52 is divisible by 4

522 is divisible by 3

5226 is divisible by 2

52267 is divisible by 1.

43268 is not reverse length divisible. Print Yes if the number is reverse length divisible and no otherwise.

**Note:**

* It is easy to solve this problem by using strings, but if you want some challenge, solve this problem without strings or any library functions.
* Be cautious about choosing the data types and the boundary conditions

**Boundary Conditions**

0<n<10000000

**Input Format**  
The first line contains the number **N**

**Output Format**  
Print "Yes" if the number is reverse length divisible otherwise print "No" (without quotes).

#### **Reverse Length Divisible**

Given a number **N,** check if it is reverse length divisible. A number is said to be reverse length divisible if the first i digits of the number is divisible by (l-i-1) where l is the number of digits in **N** and **0 < i <= l.**

For example, 52267 is reverse length divisible because:

5 is divisible by 5

52 is divisible by 4

522 is divisible by 3

5226 is divisible by 2

52267 is divisible by 1.

43268 is not reverse length divisible. Print Yes if the number is reverse length divisible and no otherwise.

**Note:**

* It is easy to solve this problem by using strings, but if you want some challenge, solve this problem without strings or any library functions.
* Be cautious about choosing the data types and the boundary conditions

**Boundary Conditions**

0<n<10000000

**Input Format**  
The first line contains the number **N**

**Output Format**  
Print "Yes" if the number is reverse length divisible otherwise print "No" (without quotes).

**ANS.**

**n=str(int(input())); flag=0**

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

**if int(n[:i])%(len(n)+1-i)!=0: break**

**else: flag=1**

**if flag==1: print("Yes")**

**else: print("No")**

Tranformation of Three Digit Numbers

**Dattatreya Ramchandra Kaprekar** (1905–1986) was an Indian recreational mathematician who described several classes of natural numbers. He observed that any three digit number with at least two different digits gets transformed to 495 while the following steps are followed:

1) Arrange the digits of number ‘n’ in ascending and then in descending order to get two four-digit numbers, m1 and m2.  
2) Compute the difference diff between m1 and m2  
3) If diff is not 495 then go to Step 1.

Given a three digit number, write a code to check if the number can be transformed to 495 following above procedure. Print the number of steps for the transformation if the number can be transformed and print No otherwise.

**Example 1**

100

5

**Explanation**

100 is transformed to 495 in five steps as shown below

100 – 001 = 099  
990 – 099 = 891  
981 – 189 = 792  
972 – 279 = 693  
963 – 369 = 594  
954 – 459 = 495

**Example 2**

222

No

**Input Format**

First line contains the number n

**Output format**

Print the number of steps taken for transformation or No

**ANS.**

**n=str(int(input())); steps=0**

**if len(set(list(n)))==1 and len(n)==3: print("No"); exit()**

**while (n!='495'):**

**steps+=1**

**step=int("".join(sorted(list(n),reverse=True)))-int("".join(sorted(list(n))))**

**if step<100: step1='0'+str(step)**

**else:step1=str(step)**

**n=step1**

**print(steps-1)**

#### **Maximum Number of Applicants from NGOs**

An MNC has a foundation which wants to provide seed fund for few Non-Governmental Organizations (NGO) committed to improve educational quality across the country. They make a call for this and circulate an online application. More than one member from a NGO can apply for the seed fund. A column in the online application ask for the number of applicants (NOA) excluding himself applying for the seed fund from the NGO to which the applicant belongs. Given ‘m’ random values of NOA column of the online application taken from few applicants of ‘k’ NGOs (where m>=k), write a code to find the maximum number of applicants from these ‘k’ NGOs.

**Example 1**

**Input**

5

7

2 3 3 3 3 1 1

**Output**

17

**Explanation**

We need to take five out of seven given values. Each different value should be belong to different NGOs, so 1,2,3 should be taken. And two more 3’s are taken. Value written in the form excludes the person who writes it so have to add 1 for each value taken

**Example 2**

**Input**

3

6

2 3 3 2 1 1

**Output**

9

**Explanation**

We need to take three out of six given values. Each different value should be belong to different NGOs, so 1,2,3 should be taken.

**Example 3**

**Input**

6

12

4 3 3 2 2 2 2 2 1 1 1 1

**Output**

19

**Explanation**

We need to take six values out of twelve given values. Each different value should be belong to different NGOs, so 1,2,3,4 should be taken. When we look at number of 1’s it is 4, for value of 1, only two enteries can be from same NGO. Therefore one more 1 that is two 1’s in total should be taken. When we look at number of 2’s it is 5, only three 2’s can be written from people of same NGO. Therefore we need to take one more 2 that is two 2’s in total. So the six values chosen are 1, 1, 2, 2, 3, 4 and one more value for each NGO should be added therefore maximum number of applicants will be 19.

**Input Format**

First line contains the number of NGO’s, k

Next line contains the number of applicants considered, m

Next line contains the NOA values written by the m applicants separated by a space

**Output Format**

Print the maximum number of applicants from k NGOs

**ANS.**

**CODEVTURTLE CODE(SEE HIS EXPLANATION AS WELL):**

**def maxcalc(dic,k,l):**

**max=0; s=0**

**for i in dic:**

**if dic[i]>=max:**

**max=dic[i]**

**s=i**

**dic[s]-=1**

**l.append(s)**

**if k!=len(l): maxcalc(dic,k,l)**

**else: return l**

**k=int(input())**

**m=int(input())**

**n=list(map(int,input().split())); dic={}**

**for i in n: dic[i]=n.count(i)**

**l=sorted([i for i in dic])**

**if len(dic)==1: print(k\*2)**

**else:**

**if len(l)==k: print(sum(l)+len(l))**

**else:**

**maxcalc(dic,k,l)**

**print(len(l)+sum(l))**

**CREATOR’S OWN SOLUTION:**

**num\_ngos=int(input())**

**num\_applicants=int(input())**

**noa\_values=list(map(int,input().split()))**

**unique\_noa=sorted(list(set(noa\_values)))**

**can\_be\_deleted={} #dictionary**

**for val in unique\_noa:**

**number\_of\_occurences=noa\_values.count(val)**

**if number\_of\_occurences%(val+1)==0:**

**possible\_ngos=(number\_of\_occurences//(val+1))**

**else:**

**possible\_ngos=(number\_of\_occurences//(val+1))+1**

**can\_be\_deleted[val]=number\_of\_occurences-possible\_ngos**

**while len(noa\_values)>num\_ngos:**

**least=unique\_noa[0]**

**if can\_be\_deleted[least]==0:**

**unique\_noa.remove(least)**

**continue**

**if can\_be\_deleted[least]==1:**

**unique\_noa.remove(least)**

**can\_be\_deleted[least]-=1**

**noa\_values.remove(least)**

**print(sum(noa\_values)+len(noa\_values))**

#### **Shortest typing time**

There are two typists : A and B. Typist A, who is experienced, can type a word in ‘a’ seconds and the Typist B can type a word in ‘b’ seconds. Given a job of typing ‘n’ number of words in a web portal, Write an algorithm and the subsequent code to compute the total time taken by both the typists to type the given ‘n’ words if they divide the job in such a manner as to complete the job in the shortest time. Assume that all the words have equal number of letters and both the typists type simultaneously.

For example, if 15 words are to be typed, typist A can type a word in 2 seconds and typist B can type a word in 3 seconds then the minimum time required to type 15 words is 18 seconds.

**Input Format**

First line contains, total number of words in the job : n

Second line contains the time taken by typist A to type a word : a seconds

Third line contains the time taken by Typist B to type a word : b seconds

**Output Format**

Minimum time required to type the words (in seconds)

**ANS.**

**from math import ceil**

**n=int(input()); a=float(input()); b=float(input())**

**print(ceil(((a\*n)\*\*(-1)+(b\*n)\*\*(-1))\*\*(-1)))**

**C++ code:**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int main(){**

**float n,a,b;**

**cin>>n>>a>>b;**

**float ans = n\*a\*b/(a+b);**

**cout<<ceil(ans);**

**}**

#### **Secondary Factorial**

Factorial of an integer n,  denoted as n! is defined as the product of the first n natural numbers

n! = 1\*2\*3....\*n

1! = 1 and 0! = 1

We define a secondary factorial of a number n, denoted by SF(n), as follows:

SF(n) = 1\*3\*5\*....\*n, if n is odd and

SF(n) = 2\*4\*6\*....\*n if n is even

If n is an odd number, SF(n) is defined as the product of all the odd numbers, starting from 1, till the number n. SF(5)= 1\*3\*5= 15.

If n is an even number, SF(n) is defined as the product of all the even numbers, starting from 2, till the number n. SF(6)=2\*4\*6=48.

Given a number k, write a code to compute SF(n), where k = n!.  For the given number k, If there is no number n such that  n! = k then, your code should print -1.

**Illstration**

Given k = 24 then 24 = 4! and

SF(4) = 2\* 4 = 8.

Given k=25, there is no number n such that 25 = n!, then the out put should be -1.

Given k=6, 6=3!. SF(3)=1\*3=3

**Input Format**

First line contains an integer, k

**Ouput Format**

Print SF(n) if there exists a number  n, such that k = n! and -1 otherwise

**ANS.**

**from math import factorial**

**def isFactorial(n) :**

**i = 1;**

**while(True) :**

**if (n % i == 0) : n//=i**

**else : break**

**i += 1**

**if (n == 1) : return 'T'**

**else : return 'F'**

**n=int(input()); x=0**

**if isFactorial(n)=='T':**

**for i in range(1,n//2+1):**

**if factorial(i)==n: x=i**

**fact=1**

**for i in range (x,0,-2): fact\*=i**

**print(fact)**

**else: print(-1)**

**SHREYASH’S code:**

**k=int(input())**

**def sec\_fact(n):**

**if n==2: return 2**

**if n==1: return 1**

**else: return n\*sec\_fact(n-2)**

**def fact(n):**

**if n==1 or n==0: return 1**

**else: return n\*fact(n-1)**

**factorial=1;i=1**

**while factorial<=k:**

**factorial=fact(i)**

**if factorial==k: print(sec\_fact(i)); exit()**

**i+=1**

**print(-1)**

**better and shorter code:**

**k,fact\_check,p = int(input()),1,0**

**while(fact\_check < k):**

**p+=1; fact\_check\*=p**

**if fact\_check == k:**

**l,sf = 0,1**

**if p%2: l = -1**

**else: l = 0**

**for i in range(p,l,-2): sf\*=i**

**print(sf)**

**else: print(-1)**

#### **Digit Cancellation Division (DCD)**

A new operation, called a Digit Cancellation Division (DCD) , represented symbolically using /// is performed among the integers. Here x///y, where x and y are integers is computed as follows.

1. Identify the common digits that occur in the integers x and y.

2. Cancel the common digits, processing from the leftmost digit to the rightmost digit. Integers with no common digits are kept as it is. If all the digits in the numerator or denominator gets cancelled then replace it by multiplicative identity 1.

3. Perform normal division between the numbers after cancellation of common digits

If y is zero, then x///y is -1.

Given two integers, write a pseudocode and the subsequent code to find x///y.

For example,

When the operation 243///41 is done, after cancellation of the common digits, we will have 23//1. The common digit 4 can occur in any position. After cancellation of the digits, 3 which was in units position , is in the ten’s position.

23///74 will be as it is.

In 1///121, we have the common digit 1, which occurs once in the numerator and twice in the denominator. In that case, digits in the left will get cancelled first.

1///121 = 1/21.

101///121 = 0/2;

2. After the cancellation of digits, perform the usual division (/)operation with the remaining digits in the numerator and the denominator.

**Input format :**

First line contains the numerator, x

Second line contains the denominator, y

**Output Format:**

Always print the value of x///y in the two decimal format.

Use format function to print two decimal places.

**Syntax:**format(s,'0.2f'), to fromat s with two decimal places

**Illustration:**

**Input :**

49

98

**Output :**

0.50

**Input :**

231

0

**Output**

-1.00

**ANS. AGAR DONO LISTS JISME KUCH BHI REMOVE YAA APPEND KARNE KA KAAM HO AUR UNN DONO LISTS KA LENGTH SAME NA HO TO PEHLE LIST KA COPY BANA LO USING THE COMMAND HIGHLIGHTED.**

**a=list(input()); b=list(input()); x=a.copy()**

**for i in x:**

**if i in b:**

**a.remove(i); b.remove(i)**

**if len(a)==0: a.append('1')**

**if len(b)==0: b.append('1')**

**if int("".join(b))==0: print('-1.00'); exit()**

**result=int("".join(a))/int("".join(b)); print(format(result,'0.2f'))**

**new code:**

**x,y = list(input())[:-1],list(input())**

**x1 = x.copy()**

**for i in x1:**

**if i in y: x.remove(i); y.remove(i)**

**if len(x) == 0: x.append('1')**

**if len(y) == 0: y.append('1')**

**if int("".join(y)) == 0: print("-1.00"); exit()**

**res = format(int("".join(x))/int("".join(y)),".2f")**

**print(res)**

#### **Maxpower**

A new operation called maxpower, among two real numbers x , y, denoted by \*\*\*, is defined as follows.

x\*\*\*y = z, where y^z =x.

x\*\*\*y is the power to which a number y must be raised in order to get x.

100\*\*\*10 = 2 , since 10 ^ 2 = 100.

2\*\*\*10 = 0.3010, since 10 ^0.3010 = 1.99 = 2 (approximately), when rounded-off to the nearest integer,

10\*\*\*2= 3.3219 since 2^3.3219 = 9.99=10 (approximately), when rounded-off to the nearest integer.

64\*\*\*4 =3, since 4^3=64.

1.44 \*\*\*1.2 = 2, since 1.2^2= 1.44

Given two real numbers x and y, write a code to print the value of x\*\*\*y with two decimal places only.

**Boundary Conditions**

x, y >= 0

**Input format:**

The first line is the number x.

The second line is the number y

**Output format**

Value of the x\*\*\*y with two decimal places only

**ANS.**

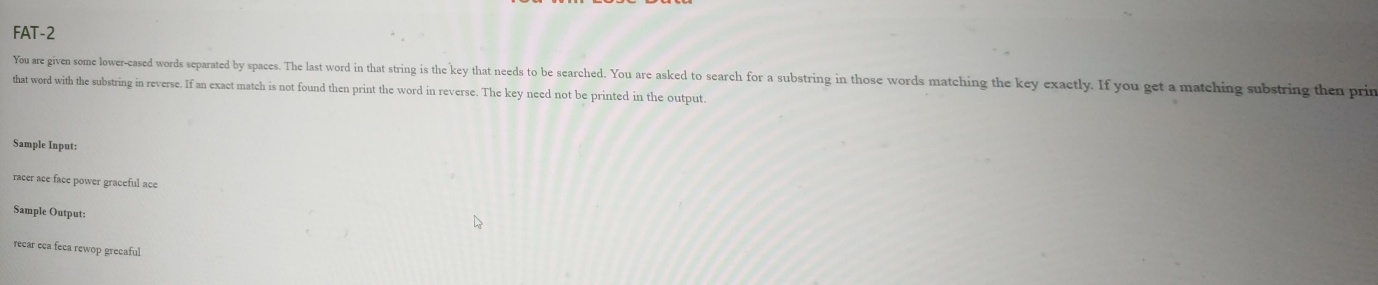
**from math import log**

**x=int(input()); y=int(input())**

**print(format(log(x,y),'0.2f'))**

**one liner code:**

**from math import log; print(format(log(int(input()),int(input())),'0.2f'))**

**Q. The ques I couldn’t do in FAT..**

**ANS.**

**2 approaches(1st one python specific and 2nd one more general):**

**s=input()**

**l=s.split(" ")**

**w=l[len(l)-1]**

**l.pop(len(l)-1)**

**removed=" ".join(l)**

**ns=removed.replace(w,w[::-1])**

**nslist=ns.split(" ")**

**nslistset=set(nslist)**

**lset=set(l)**

**common=list(set.intersection(nslistset,lset))**

**for i in range(len(nslist)):**

**if nslist[i] in common:**

**nslist[i]=nslist[i][::-1]**

**print(" ".join(nslist))**

**GENERAL ONE:**

**#final code for this question which is also optimized for the**

**#condition when a word in the string given contains multiple**

**#appearances of the key word**

**def pattern\_match(key,word):**

**ind = 0; start = None; ans = []**

**for i in range(len(word)):**

**if word[i] == key[ind]:**

**ind+=1**

**if ind == 1: start = i**

**if ind == len(key):**

**ans.append(start);**

**ind,start = 0,-1**

**else: ind = 0; start = -1**

**return ans**

**s = input().split()**

**key = s[-1]**

**s.pop()**

**for index in range(len(s)):**

**x = pattern\_match(key,s[index])**

**if len(x) != 0:**

**for j in x:**

**z = s[index][:j]+s[index][j:j+len(key)][::-1]+s[index][j+len(key):]**

**s[index] = z**

**else:**

**s[index] = s[index][::-1]**

**s = " ".join(s)**

**print(s)**

#### **Midnumeral Strings to Rightnumeral Strings**

Midnumeral string is a string which contains one number in between letters. For example, a1b2c is a midnumeral string. A midnumeral strings may contain letters a to z, numbers 1 to 9 and both left and the right parentheses.

Rightnumeral string is a string which can be formed from midnumeral string:

(i) All the characters except left and right parenthesis in the midnumeral string is present in rightnumeral string.

(ii) A number ‘n’ which is between letters ‘a’ anb ‘b’ in the midnumeral string will appear in the right hand of side of the letters ‘a’ and ‘b’ in rightnumeral string in a particular fashion.

During the process of conversion of a midnumeral string to a rightnumeral string, a waiting stream is used.

Given a midnumeral string ‘m’ it can be converted to a rightnumeral string ‘r’ as follows:

1. Process character by character of midnumeral string ‘m’.
2. When a letter is seen add it to rightnumeral string ‘r’.
3. When a number ‘n’ is seen and waiting stream is empty or the recently added character to waiting stream is left parenthesis then add ‘n’ to waiting stream
4. If a number ‘n’ seen and there are some numbers in the waiting stream then
   1. If the recently added numeral in waiting stream is less than ‘n’ then add ‘n’ to waiting stream
   2. Otherwise process waiting stream from most recent to least recent:
      1. Remove number from waiting stream and add it to ‘r’
      2. Repeat part (i), until the recent value available in waiting stream is less than ‘n’ or the recent character the waiting stream is left parenthesis or the waiting stream becomes empty
      3. Add ‘n’ to the waiting stream

c) If a left parenthesis is seen then add it to waiting stream

d) When right parenthesis is seen then remove all numbers from most recent to least recent from waiting stream and add it to r till left parenthesis is seen. Discard left and right parenthesis

4. When the end of the midnumeral ‘m’ string is reached remove all numerals from most recent to least recent from waiting stream and add it to right numeral string ‘r’

For example,

a1b2c will be abc21

a2b1c will be ab2c1

(a1b)2c will be ab1c2

a1b4(c5d2e)7f will be abcd5e2f741

**Note:**All the characters in the midnumeral will be distinct and only valid input is given

**Input Format**

First line contains the midnumeral string, m

**Output Format**

First line should contain the rightnumeral string, r

**ANS.**

**Python:**

**m,r = input(),""**

**stack = []**

**for i in m:**

**if i == '(': stack.append(i)**

**elif i == ')':**

**while(stack[-1] != '('): r+=stack.pop()**

**stack.pop()**

**elif ord(i)<=122 and ord(i)>=97: r+=i**

**elif (ord(i)>=48 and ord(i)<=57):**

**if len(stack)==0 or stack[-1]=='(': stack.append(i)**

**else:**

**while(not(len(stack) == 0 or stack[-1] == '(' or stack[-1]<i) == True):**

**r+=stack.pop()**

**stack.append(i)**

**while(len(stack) != 0): r+=stack.pop()**

**print(r)**

**C++**

**#include <iostream>**

**#include <stack>**

**#include <string>**

**using namespace std;**

**#define FOR(i,a,n) for(int i=a;i<n;i++)**

**typedef stack<char> stc;**

**int main(){**

**string m,r = "";**

**cin>>m;**

**stc wait;**

**FOR(i,0,m.length()){**

**char c = m[i];**

**if(c == '(') wait.push(c);**

**else if(c == ')'){**

**while(wait.top() != '('){**

**r+=wait.top();**

**wait.pop();**

**}**

**wait.pop();**

**}**

**else if(c<=122 && c>=97) r+=c;**

**else if(c>=48 && c<=57){**

**if(wait.empty() or wait.top()=='(') wait.push(c);**

**else{**

**while(!(wait.empty() || wait.top()=='(' || wait.top()<c)){**

**r+=wait.top(); wait.pop();**

**}**

**wait.push(c);**

**}**

**}**

**}**

**while(!wait.empty()){**

**r+=wait.top(); wait.pop();**

**}**

**cout<<r<<"\n";**

**}**

#### **Ordered-Containment of a String**

A String S2 is said to be order-contained in another string S1 if all the letters of S2 is present in S1 and order of occurrence of letters of S2 in S1 is same as in S2. For example, elephant contains ant, hat but not tap.

Given two strings S1 and S2, write a code to Print ‘Yes’ if S2 is order-contained in S1 and ‘No’ otherwise. All letters in the input will be lowercase in the given string with no spaces

**Input Format**

First line contains the string, S1

Second line contains the string, S2

**Output Format**

Print Yes if S2 is order-contained in S1 and Print No otherwise

**ANS.**

**NEW SOLUTION (O(N)):**

**s1 = input()**

**s2 = input()**

**j = 0**

**count = 0**

**for i in range(len(s1)):**

**if s1[i] == s2[j]:**

**j+=1**

**count+=1**

**if count == len(s2): print("YES"); exit()**

**print("NO")**

**better solution(optimized):**

**s1,s2 = input(),input()**

**ind = 0**

**for i in range(len(s1)):**

**if s2[ind] == s1[i]: ind+=1**

**if ind == len(s2): print("Yes"); exit()**

**print("No")**

**C++ solution(OPTIMIZED ONE):**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**#define FOR(i,a,n) for(int i=a;i<n;i++)**

**int main(){**

**string s1,s2;**

**cin>>s1>>s2;**

**int ind = 0;**

**FOR(i,0,s1.length()){**

**if(s1[i] == s2[ind]) ind++;**

**if(ind == s2.length()){**

**cout<<"Yes"<<"\n"; exit(0);**

**}**

**}**

**cout<<"No"<<"\n";**

**}**

#### **No Common Fraction (NCF)**

If a and b are positive integers, b not equal to 1, b>a, the representation a/b is called as a fraction of a to b. If the factors common to both a and b is the number 1 only, (That is, the greatest common divisor of a and b is 1), we call a/b as the `No Common Fraction’ (NCF).

Given an integer `upper’ called upper limit, an integer `lower’ called the lower limit, we can write the NCFs that can be formed with the integers that lie between u and l (lower<= integer <= upper) . For example, if upper=3, lower=1, the NCFs that can be formed, are 1/2, 1/3, 2/3.

Given the integers upper and lower, Write the pseudocode and the subsequent code to compute the number of NCFs that can be formed with the given limits and print the number of pairs whose a/b value is greater than 0.2 and less than 0.6.

For example, if upper = 3 and lower = 1 then there are two NCFs 1/3 and 1/2, satisfying the conditions.

**Input Format**

First line contains the value of upper limit, upper

Second line contains the value of lower limit, lower

**Output Format**

Print the count of NCFs that can be formed with the given upper and lower limits

**ANS.**

**CODE WHICH PASSED THE QUESTION:**

**u=int(input()); l=int(input()); a=[]**

**for i in range(l,u+1):**

**for j in range(i+1,u+1):**

**if i/j<0.6 and i/j>0.2: a.append(i/j) #set.add() function can be used to**

**print(len(set(a))) #append in sets**

**CODE WHICH WAS ACTUALLY CORRECT :**

**u=int(input()); l=int(input()); count=0**

**from math import gcd**

**for i in range(l,u+1):**

**for j in range(i+1,u+1):**

**if gcd(i,j)==1 and i/j<0.6 and i/j>0.2:**

**count+=1**

**print(count)**

#### **Circular Game**

There are N students in a class. In games period they are made to stand in a circle and each one of them is given a unique random number, r. Here r1 is the random number assigned for the first student, r2 is the number assigned for the second student and so on. In this game, for all students except those at first and last positions the winning condition is , if the student is at position ‘i’ then he has won the game if ri-1 < ri < ri+1. For the first student to win the game, the winning condition is rn<r1<r2. For the last student to win the game, winning condition is rn-1<rn<r1.

For example, if there are six students and they are assigned numbers 90, 25, 37, 28, 73, 84 then the numbers satisfying the conditions are 73, 84 and children winning the game are at positions 5 and 6.

If no one has won the game then print None. If there are six students and the numbers assigned are 35, 25, 37, 28, 84 and 73 then the output of the code should be None.

**Input Format**

First line contains the number of students in the class, n

Next n lines contain the numbers assigned for the children in order

**Output Format**

Print the position of the children who has won the game

**Note :**Position of the children are 1 to n

**ANS.**

**n=int(input()); info={}; count=0**

**for i in range(1,n+1): info[i]=int(input())**

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

**if i==1:**

**if info[i] < info[i+1] and info[i] > info[n]: print(i); count+=1**

**if i==n:**

**if info[i] < info[1] and info[i] > info[i-1]: print(i); count+=1**

**try:**

**if info[i] < info[i+1] and info[i] > info[i-1]: print(i); count+=1**

**except: pass #continue can also be used**

**if count==0: print("None")**

**better code:**

**stds,flag = [int(input()) for i in range(int(input()))],0**

**for i in range(len(stds)):**

**if i == len(stds)-1:**

**if stds[i-1]<stds[i]<stds[0]: print(i+1); flag = 1**

**else:**

**if stds[i-1]<stds[i]<stds[i+1]: print(i+1); flag = 1**

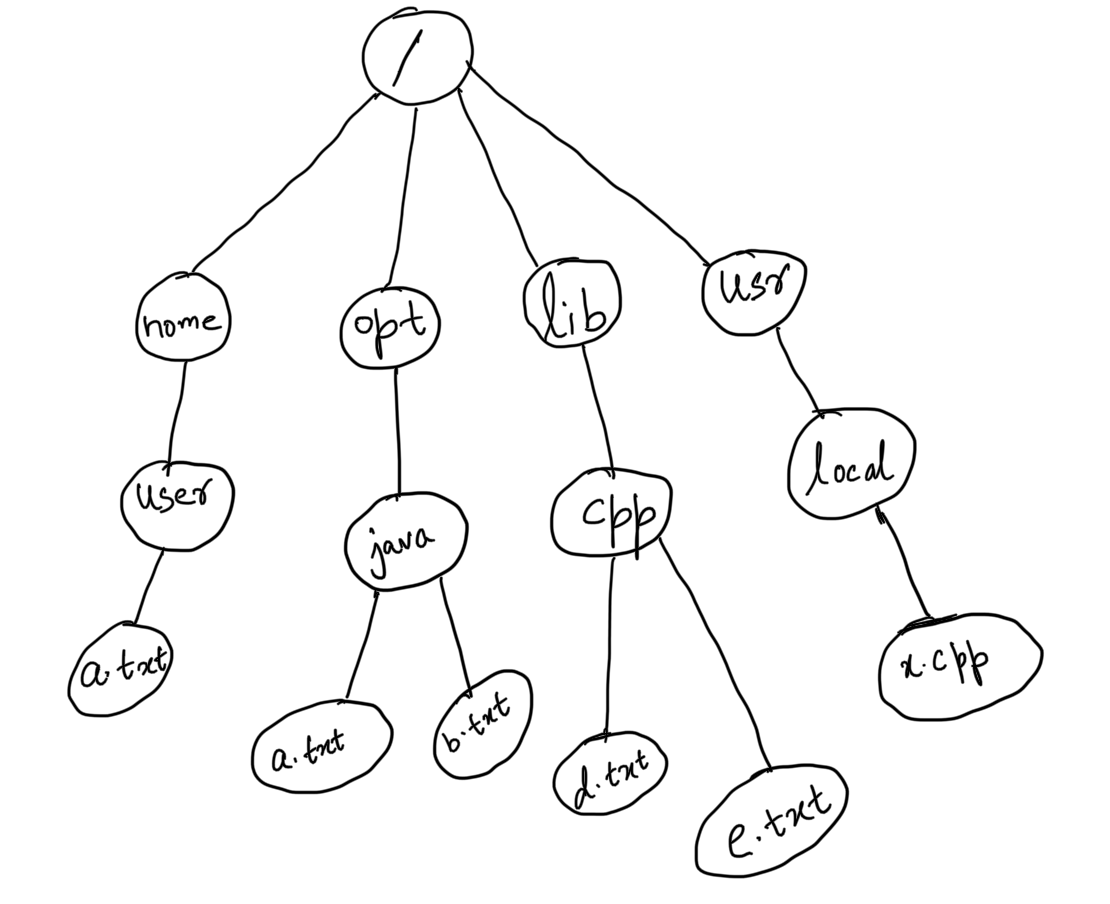
**if flag == 0: print("None")**

***File System***

A file system is a collection of files arranged in a particular order for easier access, search, editing etc. The Simple File System(SFS), has one root mount which is denoted using ‘/’.

Each file is a descendant of ‘/’. A directory is a special kind of file that can have other files organised within it. SFS also allows a user to move files from one location to another throughout the file system. SFS only allows a user to move files to a different directory only when there is no naming conflict.

For example: consider this:

****

Now consider if we have to move a file named **a.txt** from ‘**/home/user/**’ to ‘**/opt/java**’. We see that a file named a.txt already exists at the given location, so we cannot move it, however we can move ‘**/usr/local/x.cpp**’ to ‘**/home/user/**’ as there is no naming conflict. The names of all files are internally represented using IDs which are natural numbers. For example ‘/’ may have an id of 1, ‘home’ may have an id of 2 and ‘a.txt’ may have an id of 62 and so on.... each of these files have a corresponding ID. You will be asked various queries of the **<source> <destination> <filename>**and you have to tell **‘yes’**or **‘no’**(without quotes) if the files can be moved from source to destination or not.

Consider the above example:

Given the queries of the form:

1. /home/user /home a.txt
2. /home/user /opt/java a.txt
3. /lib/cpp /usr/local d.txt

1. For the first one we are moving **a.txt** from **/home/user**to **/home** which is possible as there is no naming conflict, so we say yes.
2. We are moving **a.txt**from **/home/user to /opt/java .**A file named a.txt already exists in the destination hence, we say no.
3. We are moving **d.txt**from **/lib/cpp**to **/usr/local.**As no file exists with the same name in destination we say yes.

**Note:**

(i) There are no duplicates in directory but duplicates are there in files

(ii) All file names have an extension separated by a dot '.' and directories do not have extensions

(iii) It is guaranteed that the path given will be valid and the files with given names in the queries will be present and source and destination will not be same.

**Input format:**

First line contains a single integer denoting the number of files, **N**

Second line contains a single integer denoting the edges in the file system, **M**

Next **M**lines contain two natural numbers separated by a space denoting the ids of the files joined by the edge.

Next line contains **N**space separated strings and the names of all the files.

Next line contains a positive integer **Q**denoting the number of queries.

Next **Q**lines contain queries of the form <string> <destination> <filename>, each separated by a space.

**Output format:**

Output **Q**lines each containing a single string **‘yes’**or **‘no’** (without quotes), if the file can be moved or not.

**Example:**

**Input:**

15

14

1 2

1 3

1 4

1 5

2 6

3 7

4 8

5 9

6 10

7 11

7 12

8 13

8 14

9 15

/ home opt lib usr user java cpp local a.txt a.txt b.txt d.txt e.txt x.cpp

3

/home/user /home a.txt

/home/user /opt/java a.txt

/lib/cpp /usr/local d.txt

**Output:**

yes

no

yes

**Explanation:**

IDs of the files are as:

/ - 1

home - 2

opt - 3 and so on

The rest of the explanation can be taken from the example given in the problem

TEST CASE 1:

Input

15

14

1 2

1 3

1 4

1 5

2 6

3 7

4 8

5 9

6 10

7 11

7 12

8 13

8 14

9 15

/ home opt lib usr user java cpp local a.txt a.txt b.txt d.txt e.txt x.cpp

5

/home/user /home a.txt

/home/user /opt/java a.txt

/lib/cpp /usr/local d.txt

/home/user / a.txt

/usr/local / x.cpp

Expected output

yes

no

yes

yes

yes

TEST CASE 2:

Input

15

14

1 2

1 3

1 4

1 5

2 6

3 7

4 8

5 9

6 10

7 11

7 12

8 13

8 14

9 15

/ home opt lib usr user java cpp local a.txt a.txt b.txt d.txt e.txt x.cpp

3

/home/user /home a.txt

/home/user /opt/java a.txt

/lib/cpp /usr/local d.txt

Expected output

yes

no

yes

**ANS. YET TO BE FOUND**

#### **Scanner**

A scanner reads an image with 10 lines. Each line is given a value from 0 to 9 based on their width as follows:

0.2 to 0.7 – 0

0.71 to 1.2 - 1

1.21 to 1.7 – 2

1.71 to 2.2 – 3

2.21 to 2.7 – 4

2.71 to 3.2 – 5

3.21 to 3.7 – 6

3.71 to 4.2 – 7

4.21 to 4.7 – 8

4.71 to 5.2 – 9

There may be errors in the value read by the scanner. So always a check digit is given to verify the correctness. Check digit of the number scanned can be calculated as modulus of sum of product of the digits and weightage by 10. Weightage of the digits are given alternatively 1 and 3 from left to right. Given width of the ten lines scanned by the scanner and the cehck digit, write a code to check if it is read correctly. Print Yes if it is correct and No otherwise.

For example,

**Width of lines -** 4.2 3.5 2.8 1.5 2.2 4.7 5.0 0.5 1.1 3.8

**Digits -**7 6 5 2 3 8 9 0 1 7

**Weightage -** 1 3 1 3 1 3 1 3 1 3

Sum of product digits and the corressponding weights is 94. Modulus of 10 will be 4. Check digit is 4 so print Yes

**Note:**

1) In C language, real constants are internally considered as double values, i.e. 2.5 is a double value

2) So if you want to check if a floating point variable f is equal to 0.5 then you have to write if (f==0.5f), add a suffix f to force 0.5 to be float value. Otherwise there will be inconsistency in the output

**Input format**

First line contains the width of the ten lines read by the scanner

Next line contains the check digit

**Output Format**

Print Yes if check digit calculated and given are same and No otherwise

**ANS.**

**l=list(map(float,input().split())); check=int(input()); digit=[]; weightage=[]**

**sum=0**

**for i in l:**

**if i>=0.2 and i<=0.7: digit.append(0)**

**elif i>=0.71 and i<=1.2: digit.append(1)**

**elif i>=1.21 and i<=1.7: digit.append(2)**

**elif i>=1.71 and i<=2.2: digit.append(3)**

**elif i>=2.21 and i<=2.7: digit.append(4)**

**elif i>=2.71 and i<=3.2: digit.append(5)**

**elif i>=3.21 and i<=3.7: digit.append(6)**

**elif i>=3.71 and i<=4.2: digit.append(7)**

**elif i>=4.21 and i<=4.7: digit.append(8)**

**elif i>=4.71 and i<=5.2: digit.append(9)**

**for i in range(len(digit)):**

**if i%2==0: weightage.append(1)**

**else: weightage.append(3)**

**for i in range(len(digit)):**

**sum+=digit[i]\*weightage[i]**

**result=sum%10**

**if result==check: print("Yes")**

**else: print("No")**

#### **Sliding Friction**

Sliding friction is contact force generated when an object moves over a surface. Few wooden blocks of different heights are put in a room. An experiment is to be carried over in which a ball has to roll up and down over a steps like structure. The experimenter wishes to find the maximum size of the steps structure present in the wooden block and he prefers to have a step up operation from left to right. Given the height of the wooden blocks arranged in the room, write a code to find the starting and ending position of the wooden blocks that can form a steps structure. When more than one steps structure of same maximum size is available consider the one that comes first from left to right.

**Example 1**

Heights =5 6 3 5 7 8 9 1 2

Longest step up structure from left to right is

3 5 7 8 9

and positions are from 3 to 7

Heights = 12 13 1 5 4 7 8 10 10 11

Longest step up structure from left to right is

4 7 8 10

and positions are from 5 to 8

**Input Format**

First line contains the number of wooden blocks, n

Next line contains the height of the wooden blocks separated by a space

**Output Format**

Print the starting and ending positions of the wooden blocks that can form a step structure of maximum size separated by a space

**ANS.**

**Better code in C++**

**#include <iostream>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**int max\_start = 1,max\_end = 1,max\_len = 1;**

**int prev\_height = -1,start = 1,end = 1,len = 1;**

**for(int i=1;i<n+1;i++){**

**int curr\_height;**

**cin>>curr\_height;**

**if(i == 1){**

**prev\_height = curr\_height;**

**continue;**

**}**

**else{**

**if(prev\_height < curr\_height){**

**end++; len++;**

**}**

**else{**

**if(max\_len < len){**

**max\_len = len;**

**max\_start = start;**

**max\_end = end;**

**}**

**start = i; end = i;**

**len = 1;**

**}**

**prev\_height = curr\_height;**

**}**

**}**

**cout<<max\_start<<" "<<max\_end;**

**}**

**n=int(input())**

**l=list(map(int,input().split()))**

**indexes=[]**

**a=set()**

**for i in range(n-1):**

**if l[i+1]>l[i]:**

**a.add(i+1) #hume sirf indexes chaie the step up waale seq. mein elements ke**

**a.add(i+2)**

**continue #yeh isliye kiya taaki yeh condition aage tak check ho jabtak false nahi ho jaati**

**#nahi toh pair mein add krdega sub-sets mein of lists.**

**indexes.append(a)**

**a=set() # agar sets ko lists ke andar use karna hai toh aise krna padega**

**pos1=min(max(indexes,key=len)) #list ke andar koi aur attribute mein bada ya chota value #chaie toh aise**

**pos2=max(max(indexes,key=len))**

**print(pos1,pos2)**

#### **Simple Tetris**

The game of tetris involves blocks of various colours dropping on top of each other If four blocks of the same colour are stacked on top of each other then they disappear, otherwise the height of the tower keeps on growing. You will be given two queries:

**Type 1:**Add a block of given colour on the tetris stack.

**Type 2:**Tell the current height of the tetris stack.

**Input Format:**

The first line of the input contains a single positive integer **Q,**which is the number of queries.

Then **Q** lines follow, each containing either of the two types of query:

**Type 1 query format:**1 <colour>, Two space separated values, colour is a string of length not more than 20

**Type 2 query format:**2

**Example**

**Sample Input:**

10

1 Green

1 Red

1 Red

2

1 Green

1 Green

2

1 Green

1 Green

2

**Sample Output:**

3

5

3

**Explanation:**

First we get

1 Green

1 Red

1 Red

We have 3 boxes stacked on top of each other and now we are asked the height, its 3.

Moving on, we stack two more green boxes and again we are asked the height. At this point, the height is 3 + 2 = 5

Now again two type 1 queries adding the green boxes to the tetris tower and now height is asked. Since 4 continuous green colored boxes are stacked on top of each other, they disappear and the height again becomes 3.

**ANS.**

**#include <iostream>**

**#include <stack>**

**using namespace std;**

**int main(){**

**int q; cin>>q;**

**stack <string> tetris;**

**string prev;**

**int consec = 1;**

**for(int i=0;i<q;i++){**

**int type;**

**string color;**

**cin>>type;**

**if(type == 1){**

**cin>>color;**

**tetris.push(color);**

**if(prev == color){**

**consec++;**

**if(consec == 4){**

**while(consec > 0){**

**consec--;**

**tetris.pop();**

**}**

**}**

**}**

**else consec = 1;**

**prev = color;**

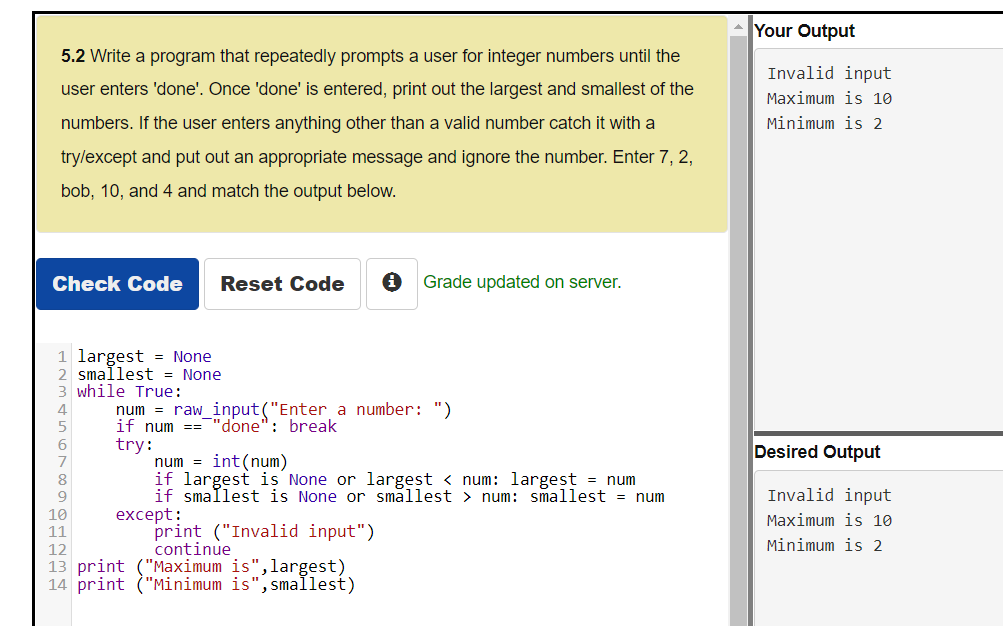
**}**

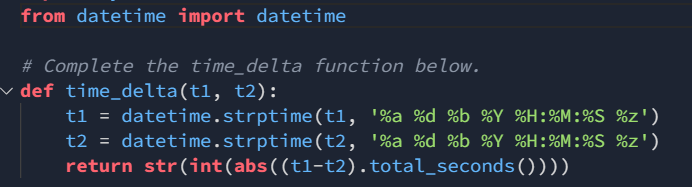
**else cout<<tetris.size()<<"\n";**

**}**

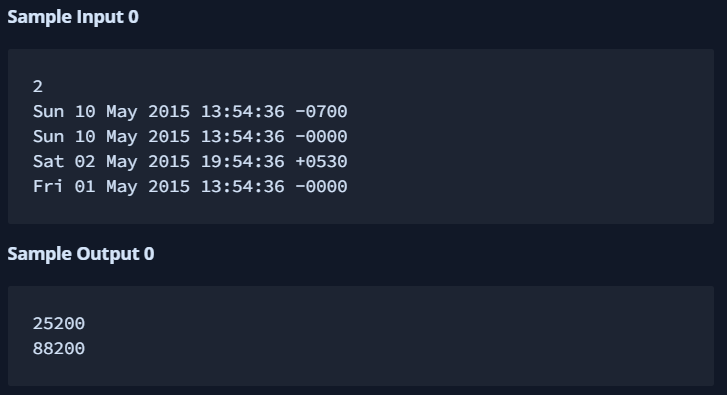
**}**

**COURSERA ASSIGNMENT QUES. :**

****

**COMPUTING THE TIME DIFFERENCE USING PYTHON BETWEEN TWO TIMES IN DIFFERENT TIME ZONES AND DAY DATE ALL GIVEN :**

**OUTPUT SAMPLE:**

****

#### **A Magic Key and a Gift**

An angel gave box with ‘N’ pearls in it and a magic key ‘K’ to a peasant. The angel asked the peasant to apply the key repeatedly till the box gets opened.

The box will get opened only when it becomes empty and the key performs a magic for each time the peasant applies it to the box. Each time when the key is used on the magic box, the following things happen:

(i) Pearls in the box will be grouped into ‘G’ groups such that each group contains exactly ‘K’ pearls.

(ii) The magic box piles up the pearls left ungrouped into a safety locker inside it

(iii) After piling up, the box will contain lesser pearls equal to the number of groups ‘G’ formed in the current step

When the box becomes empty the box opens with the pearls in the safety locker and given as gift to the peasant.

Given the number of pearls, ‘N’ and the key ‘K’, write a C program to display the number pearls that got saved in the locker from the recent time to the initial time and the total number of pearls that is given as gift for the peasant.

For example, if ‘N’ is 200 and ‘K’ is 7, then the locker displays the number of pearls saved from recent time to the initial time as 404 and the peasant gets 8 pearls as his gift.

**Input format:**

First line contains the number of pearls ‘N’ in the magic box

Next line contains the key value ‘K’

**Output format:**

In the frist line, print the number of pearls saved from recent time to the initial time in the same line continuously

In the next line print the number of pearls given as gift for the peasant

**ANS.**

**PYTHON CODE:**

**n=int(input())**

**k=int(input())**

**sum=0**

**piled\_up=[]**

**while n!=0:**

**g=n//k**

**r=n%k**

**piled\_up.append(str(r))**

**sum+=r**

**n=g**

**print("".join(reversed(piled\_up)))**

**print(sum)**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main()**

**{ int n,k,sum=0,g,r;**

**vector <int> piled\_up;**

**cin>>n;**

**cin>>k;**

**while(n!=0)**

**{**

**g=int(n/k);**

**r=n%k;**

**piled\_up.push\_back(r);**

**sum+=r;**

**n=g;**

**}**

**for(int i=piled\_up.size()-1;i>=0;i--)**

**{**

**cout<<piled\_up[i];**

**}**

**cout<<endl<<sum;**

**}**

#### **MATRICES UNDERSTANDING PROBLEM:**

#### **Enlarge Move in Gain-Coins Game**

Gain-Coins is a mxn board game, in which the number of coins in each square increases during enlarge move. The number of coins on each square ‘s’ which is row ‘r’ and column ‘c’ increases by the maximum number of coins in the row ‘r’ and also increases by the minimum number of coins in the column of ‘c’. Given the number of coins in the board write a C program to print the number of coins in the board after enlarge move. For example if the number of coins in a 3X3 board is:

2 3 4

5 7 3

3 8 1

Then after enlarge move the number of coins will be:

8 10 9

14 17 11

13 19 10

**Input Format**

First line contains the number of rows in the board, r

Next line contains the number of columns in the board, c

Next ‘r’ lines contain the number of coins on each row separated by a space

**Output Format**

Print the number of coins on the board after enlarge operation

Print the number of coins in one row on each line separated by a space

**Note:**A space is present at the end of each row

**ANS.**

**# inputting values**

**r=int(input()); c=int(input()); board=[]**

**for i in range(r): board.append(list(map(int,input().split())))**

**# short form max\_r values**

**max\_r=list(map(max,board))**

**# for min\_c values**

**min\_c=[]**

**for i in range(c):**

**column=[]**

**for j in range(r):**

**column.append(board[j][i])**

**#print(j,i,end=" ")**

**#we can get column elements together by reversing the indices in iterator variable**

**min\_c.append(min(column))**

**# addition operation on each element**

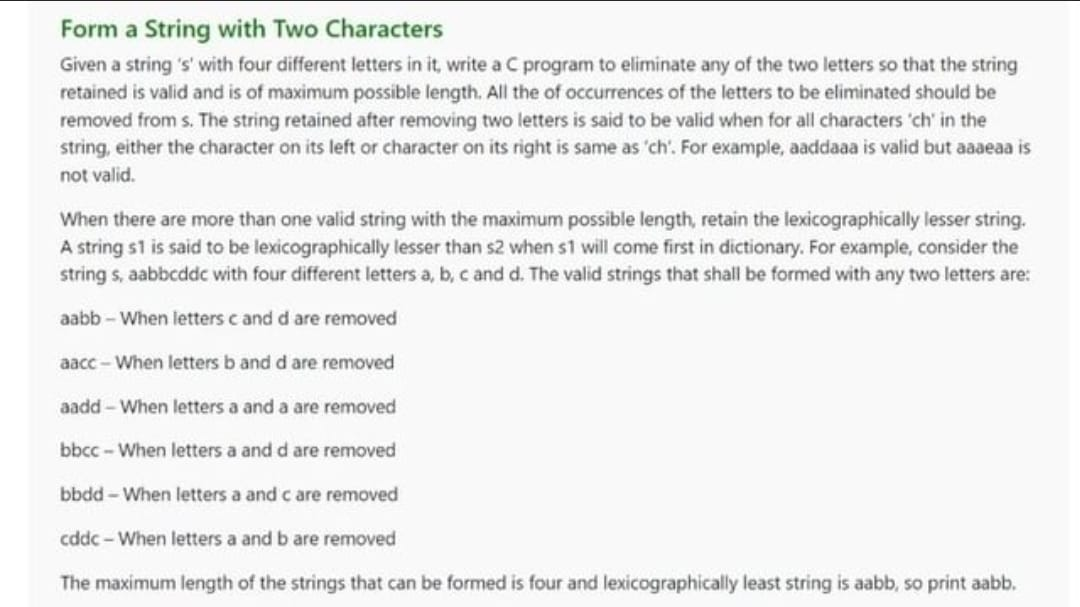
**for i in range(r):**

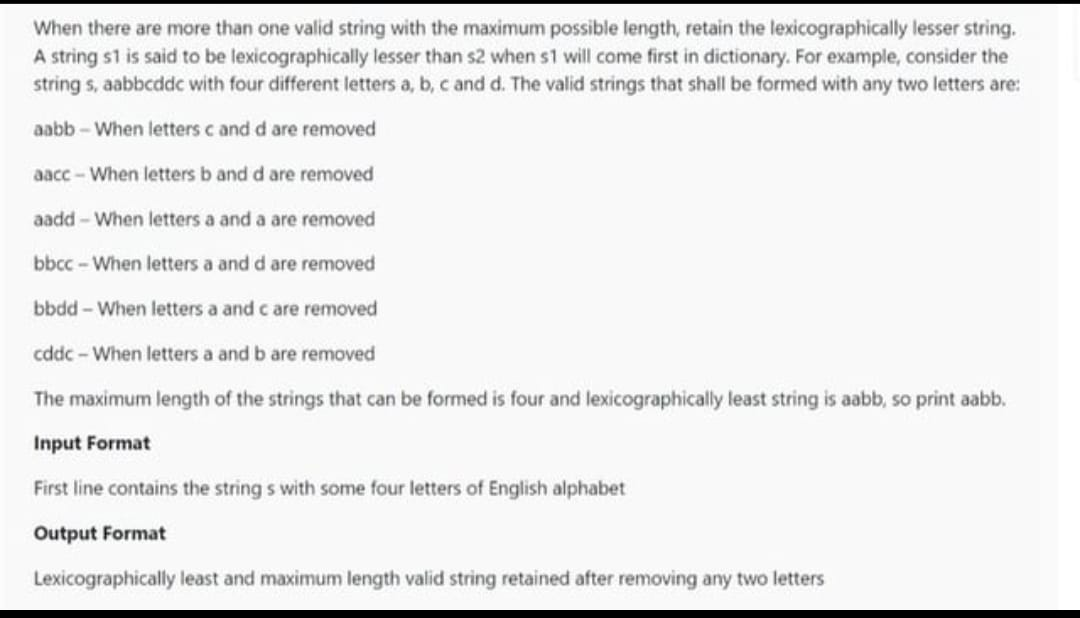
**for j in range(c): #this happens different**

**board[i][j]+=(max\_r[i]+min\_c[j])**

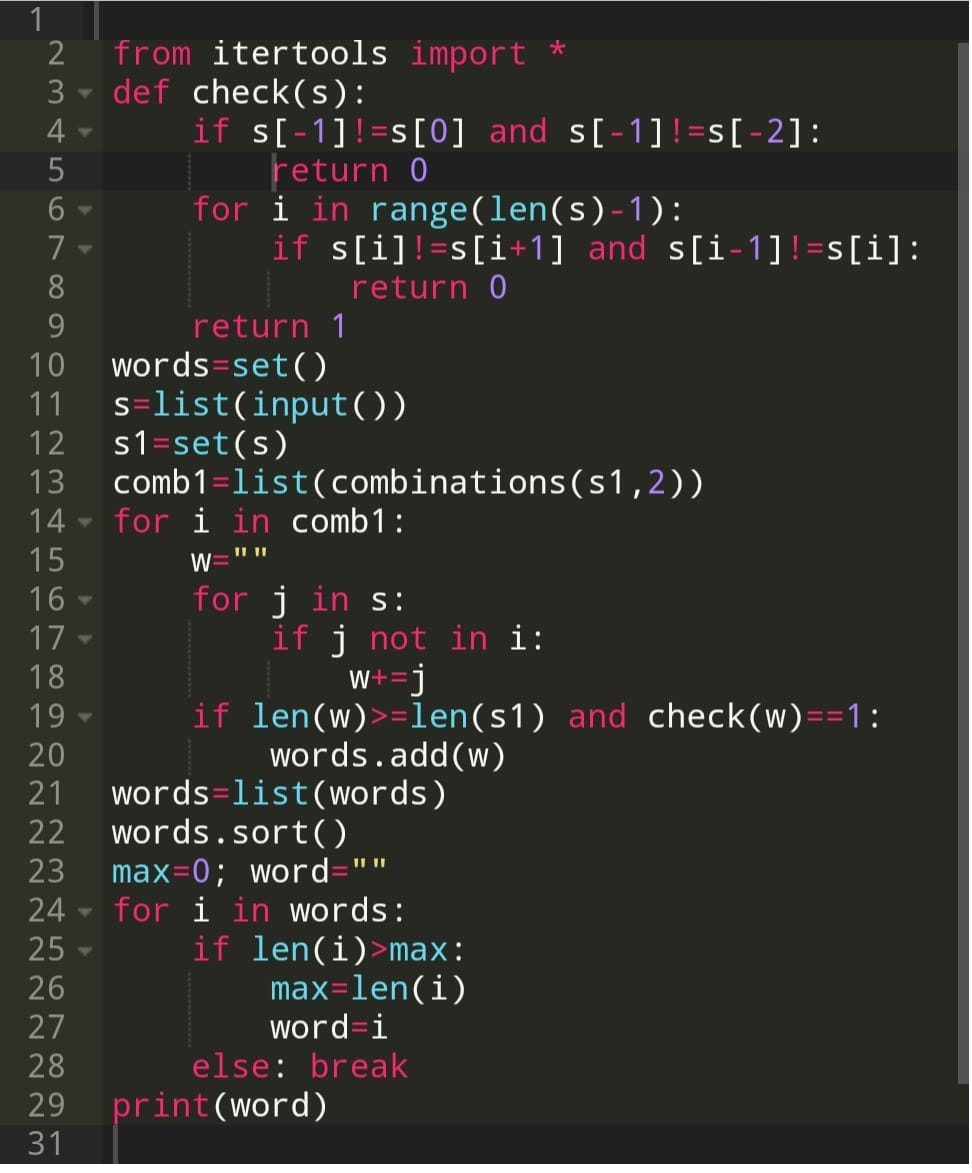
**print(\*board[i],"") # for printing in the same for loop this happens diff.**

**Q.**

****

****

**ANS.**

****

#### **Monkey Jumps Till Zero or Loop**

A big hall’s floor is filled linearlly with ‘n’ rectangular cardboards of same size ‘s’. Each cardboard is given an index ‘i’ indicating the position of it on the floor. The value of ‘i’ ranges from 1 to ‘n’. And some integer ‘p’ is written on each cardboard. n-1 values written on the cardboard are continuous integers starting from some number ‘m’ and one of the value is repeated.

A monkey enters the hall and sits on the first cardboard and then it finds the difference ‘d’ between the numbers written on the carboard on which it is sitting and the adjacent cardboard on the right side. Then it jumps to the ‘dth’ carboard. It continues this either till the difference ‘d’ is zero or when it sits on the same cardboard which it has previously sat. If ‘d’ is zero then the monkey is happy and when the monkey has to sit on the cardboard on which it sat already it gets angry. In both the cases the monkey quits the hall.

Given n integers written on the cardboards arranged in the order from 1 to n, write a C program to print the numbers written on the cardboards on which the monkey sits. Also print either Happy or Angry based on the condition the monkey quits.

For example, given eight integers written on the cardboard as 34, 39, 33, 37, 37, 35, 38, 36 your program should print 34, 37, 39, 35, 33, and 37 and also print Happy. If six integers are written in the cardboard as 34, 36, 35, 35, 38, 37 then the monkey will sit on cardboard with numbers 34, 36, 34 sees that it has sat on the cardboard which it had sat already so gets angry. So print 34, 36, 34 and Angry.

**Boundary Condition**

n<20

**Input Format**

First line contains the number of cardboards, n

Next line contains ‘n’ numbers written on the cardboard separated by a space

**Output Format**

Print the numbers on the cardboard that the monkey sits in sequence separated by a space. Print Happy or Angry based on the quit condition of monkey

**ANS.**

**n=int(input())**

**nums=list(map(int,input().split()))**

**d=None; sit=[]; k=0**

**while d!=0:**

**if nums[k] not in sit:**

**sit.append(nums[k])**

**d=abs(nums[k+1]-nums[k])**

**k=d-1**

**else: sit.append(nums[k]); break**

**print(\*sit,"")**

**if sit[-1]==sit[0]: print("Angry")**

**else: print("Happy")**

**C++ ANSWER (USING STL):**

**#include <iostream>**

**#include <algorithm>**

**#include <vector>**

**using namespace std;**

**int main()**

**{**

**int d,k=0,n;**

**cin>>n;**

**int nums[n];**

**vector <int> sit;**

**cout<<endl;**

**for (int i=0;i<n;i++) {cin>>nums[i];}**

**while(d!=0)**

**{ if (find(sit.begin(),sit.end(),nums[k])==sit.end())**

**//find function is used to find an element in a vector it stops at the index at //which it finds the value given as argument (found in algorithm header file)**

**{**

**sit.push\_back(nums[k]);**

**d=abs(nums[k+1]-nums[k]);**

**k=d-1;**

**}**

**else**

**{**

**sit.push\_back(nums[k]);**

**break;**

**}**

**}**

**for (int i=0;i<sit.size();i++) {cout<<" "<<sit[i];} cout<<" ";**

**if(sit[sit.size()-1]==sit[0]) { cout<<endl<<"Angry";}**

**else{cout<<endl<<"Happy";}**

**}**

#### **Simple String Subtraction**

Given two strings S1 and S2, write a C program to perform simple string subtraction. Simple string subtraction does character subtraction from last to first characters of the strings. Letters in English alphabet are given forward and reverse positions 1 to 26. In forward position letter ‘a’ is at 1 and ‘z’ is at 26. In reverse position letter ‘z’ is at 1 and letter ‘a’ is at 26. Character subtraction of character C2 from character C1 (C1 - C2) works as follows:

(i) If C1 and C2 are same then C1 – C2 = C1

(ii) Otherwise C1 – C2 = C3 where C3 is the character in forward position ‘d’ where d = i – j, i and j are the foward position of C1 and C2 respectively and d>0 and

C1 – C2 = C3 where C3 is the character in reverse position ‘d’ where d = positive value of (i – j), i and j are the foward position of C1 and C2 respectively and d<0.

For example, if S1 is elephant and S2 is apple then S3 is eleoslbo and if S1 is ball and S2 is cell then S3 is zwll

**Boundary Conditions**

The input will be given such that length of S1 is always greater than or equal to length of S2 and all characters in S1 and S2 are in lower case.

**Input Format**

First line contains the string S1

Next line contains the string S2

**Output Format**

String obtained after simple string subtraction

**ANS.**

**s1=list(input()); s2=list(input()); s3=""; letters="abcdefghijklmnopqrstuvwxyz"**

**for i in range(len(s1)):**

**if len(s2)-i-1>-1:**

**if s1[len(s1)-i-1]==s2[len(s2)-1-i]: pass**

**else:**

**d=letters.index(s1[len(s1)-i-1])-letters.index(s2[len(s2)-i-1])**

**if d>0: s1[len(s1)-i-1]=letters[d-1]**

**else: s1[len(s1)-i-1]=letters[d]**

**print("".join(s1))**

**# alternate : we can put \*\*[ elif d==0: pass ]\*\*condition inside the first if and remove the**

**# existing separate condition but the condn. written in the program is faster #for execution**

#### **Longest ordered vowel sequence**

A word ‘W’ is said to be an ordered vowel sequence if the vowels in ‘W’ are ordered [‘a’, ‘e’, ‘i’, ‘o’, ‘u’]. For example, if the word is ‘apple’ then it is an ordered vowel sequence whereas the word ‘elephant’ is not. An ordered vowel sequence ‘S’ is said to be the longest, if the count of occurrences of unique vowels is the highest. For example, the count of occurrences of unique vowels in ‘captain’ is 2.

Given a line of text, develop an algorithm and C program to display the longest vowel sequence. If there is more than one word being the longest, then all the words should be printed in the order of their occurrence.

For example, if the given text is ‘a tailor has become a sailor after he has experienced a voyage with his captain friend’, then the longest unique vowel sequences are ‘tailor’ and ‘sailor’.

Note: The text contains lowercase letters. To get a line of text in a variable str, use the following code snippet:

scanf(“%[^\n]s”, str);

**Input format:**

Enter a line of text

**Output format:**

Print the longest unique vowel sequences. If there is more than one such sequence, then print each one in new line in the order of their occurrence

**Note:**There was a mistake in hidden test cases. Now it is modified. Apologies for the mistake

**ANS.**

**s=input().split(); vowels="aeiou"; info={}**

**for i in s:**

**unique\_word\_vowels=[]**

**for j in i:**

**if j in vowels and j not in unique\_word\_vowels:**

**unique\_word\_vowels.append(j)**

**if "".join(sorted(unique\_word\_vowels))=="".join(unique\_word\_vowels):**

**info[i]=len(unique\_word\_vowels)**

**else: pass #can be omitted but put here for understanding**

**x=max(info.values())**

**for i in info.keys():**

**if info[i]==x : print(i)**

#### **Underscored PascalCase**

PascalCase is a naming convention of variables in programming language in which a variable name is formed from multiple words that are joined together as a single word with the first letter of each of the multiple words capitalized. Underscored PascalCase is an extension of PascalCase where an underscore is introduced between each word. Given ‘n’ words write a C program to form underscored PascalCase name using the given ‘n’ words. For example, if five words counter, for, looping, array, elements then the output should be Counter\_For\_Looping\_Array\_Elements.

**Note:**

(i) Strcat is a built-in function in C (declared in string.h) that takes two arguments, i.e, two strings and stores the resultant concatenated string in the first string specified as first argument to the function.

(ii) It is important to remember that C does not initialize memory and we have to initialize it whenever required and we will get unexpected results otherwise

**Boundary Conditions**

0<n<=10

Length of each word < 20

**Input Format**

First line contains the number of Words

Next ‘n’ lines contain words

**Output Format**

Print the Underscored PascalCase name formed with the given words

**ANS.**

**PYTHON:**

**from string import \***

**words=[input().capitalize() for i in range(int(input()))]**

**print("\_".join(words))**

**C PROGRAM: done using array of strings:**

**#include <stdio.h>**

**#include <ctype.h>**

**int main(){**

**int n;**

**scanf("%d",&n);**

**char words[n][20];**

**for(int i=0;i<n;i++){**

**scanf("%s",words[i]);**

**words[i][0]=toupper(words[i][0]); //subs. – char(int(words[i][0])-32)**

**}**

**for (int i=0;i<n;i++){**

**if(i!=n-1){printf("%s\_",words[i]);}**

**else {printf("%s",words[i]);}**

**}**

**}**

**C++ PROGRAM: done using vector of strings**

**can be done using array of strings in C++ as well**

**#include <iostream>**

**#include <string>**

**#include <vector>**

**#include <cctype>**

**using namespace std;**

**int main(){**

**int n;**

**cin>>n;**

**vector <string> words;**

**for (int i=0;i<n;i++){**

**string word;**

**cin>>word;**

**word[0]=toupper(word[0]);**

**words.push\_back(word);**

**}**

**for (int i=0;i<n;i++){**

**if (i!=n-1) {cout<<words[i]<<"\_";}**

**else {cout<<words[i];}**

**}**

**}**

#### **Find Abundant Mineral**

Many minerals such as olivine, quartz, feldspar, mica, pyroxene and amphibole are found in rocks. It has been observed that in Madhya Pradesh, only one of the mineral occurs abundantly. Here the mineral that occurs in maximum number of rocks is said to be abundant. Given the details of minerals found in ‘n’ rocks write a C program to find out the mineral that has occured maximum number of times. Print only the representation of the mineral that has occurred maximum number of times. Each mineral is represented by one letter that is for example, olivine will be represented as o, quartz as q and so on. The minerals found in each rock is given as a string.

For example, given the details of minerals in four rocks as oqm, qfo, mpa and opf the representation of the mineral that has occurred abundantly is o.

**Boundary Conditions**

0<n<20

0<number of minerals<20

**Input Format**

First line contains the number of rocks, n

Next ‘n’ lines contain the details of minerals in each rock

**Output Format**

Print the representation of mineral that occur abundant

**ANS.**

**from collections import Counter**

**s="".join([input() for i in range(int(input()))])**

**letters=list(Counter(s).keys()); freqs=list(Counter(s).values())**

**print(letters[freqs.index(max(freqs))])**

#### **Print the Pattern**

Given the value of ‘n’ write a C program to print a special rectangular pattern with dots and starts. When the value of n is 5 the rectangle looks as below:

. . . . \* . . . .

. . . \* . \* . . .

. . \* . . . \* . .

. \* . . . . . \* .

\* . . . . . . . \*

**Input Format**

First line contains the value of ‘n’

**Output Format**

Print the pattern appropriate to the value of ‘n’

Dots and stars are separated by one space and each row in the pattern is separated only by a new line character.

**ANS.**

**#include <stdio.h>**

**int main(){**

**int n;**

**scanf("%d\n",&n);**

**int k=n;**

**for (int i = n;i>=1;i--)**

**{**

**for (int j = 1;j<2\*n-1;j++)**

**{**

**if (j==i || j==k)**

**printf("\* ");**

**else**

**printf(". ");**

**}**

**k++;**

**if(i==1)**

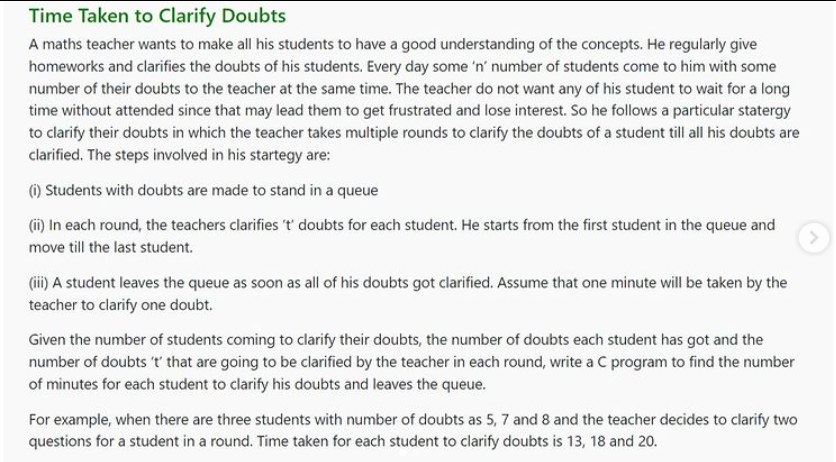
**printf("\*");**

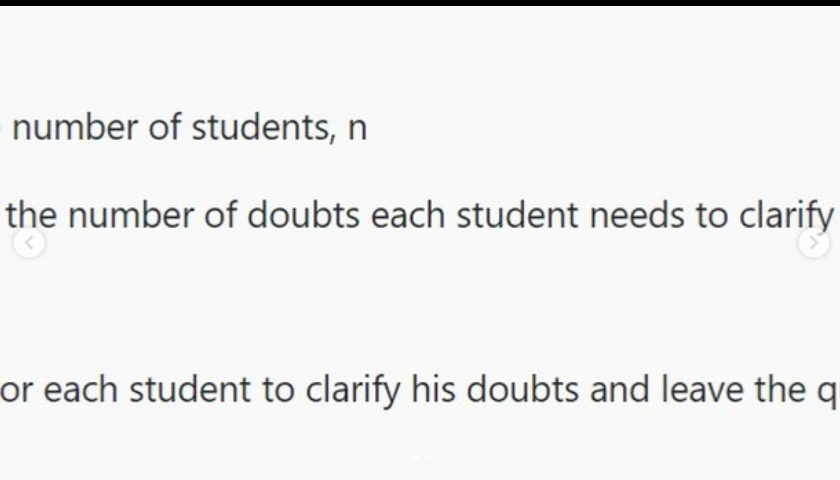
**else**

**printf(".\n");**

**}**

**}**

****

****

**Input:**

**n, list of students with the number of doubts of each and t(no. of doubts teacher solves at each round)**

**ANS.**

#### **Bit Masking**

Bit masking is an important phenomenon which is very much helpful in graphical applications. In bit masking, bitwise operations are used to on, off, toggle and query status of bits. Bitwise OR operation with ‘1’ sets a bit on and bitwise OR with ‘0’ leaves a bit unchanged. Bitwise AND operation with a ‘0’ leaves it off and bitwsie AND with ‘1’ leave it unchanged. To toggle a bit, that is to change the bit from ‘0’ to ‘1’ or vice versa it has to be XOR-ed with ‘1’ or ‘0’ as required.

Querying a bit can be acheived using AND operation, that is to know the status of the least significant bit (right most) of a number ‘x’, AND operation can be perfomed with 1. If the least significant bit is 1 then the result of the bitwise AND operation is 1 and it is 0 otherwise. Similarly ‘x’ AND 2 will return 2 when the second bit from right is 1 and 0 otherwise. To check the third, fourth bit and so on bitwise AND operation may be performed with numbers 4, 8, 16 and so on.

Given two unsigned integers ‘M’ and ‘N’, write a C program to find out the bit positions with same bits. For example, when M is 268435446 and N is 1006, unsinged integers a re represented using 4 bytes that is 32 bits internally and representation of these numbers internally are:

M - 00001111111111111111111111110110

N - 0000000000000000000001111101110

Number of bits positions with same bits is 12 and they are:

1, 2, 3, 6, 7, 8, 9, 10, 29, 30, 31, 32

Bit positions are numbered 1 to 32 from right to left.

**Input Format**

First line contains the value of M

Next line contains the value of N

**Output Format**

Print the bit positions with same bits on a line and separated by a space

**ANS.**

**def len\_changer(n):**

**if len(n)<32:**

**s=""**

**for i in range(32-len(n)):**

**s+="0"**

**b=s+n**

**return b**

**else: return n**

**m=bin(int(input()))[2:] ; n=bin(int(input()))[2:] ; b="" ; b1=""**

**b=len\_changer(n) ; b1=len\_changer(m); pos=[] ; count=0**

**for i in range(len(b)):**

**if int(b[i])==int(b1[i]):**

**count+=1**

**pos.append(32-i)**

**print(\*reversed(pos))**

**C++**

**#include <iostream>**

**using namespace std;**

**int main(){**

**int m,n;**

**cin>>m>>n;**

**int m1 = m,n1 = n;**

**for(int i=0;i<32;i++){**

**int x = (m1>>i)&1,y = (n1>>i)&1; //this operation returns i th set bit**

**if(x == y) cout<<i+1<<" ";**

**m1 = m; n1 = n; //but requires resetting the number**

**}**

**}**

#### **Shuffled Rows Matrix**

A matrix is said to be shuffled rows matrix if the elements in all the rows are same but may be in a shuffled order. Given nxm matrix, write a C program to check if the matrix is a Shuffled Rows Matrix. For example, a 3x4 matrix M as shown below is a Shuffled rows matrix:

4 5 1 7

7 1 4 5

1 7 5 4

Whereas the matrix M1 shown below is not a Shuffled row matrix

3 1 2

3 3 1

1 2 3

**Boundary conditions**

0<r,c<20

**Input Format**

First line contains the number of rows in the matrix, m

Next line contains the number of columns in the matrix, n

Next ‘m’ lines contain the elements in each row of the matrix separated by a space

**Output Format**

Print Shuffled Row Martix or Not Shuffled Row Martix

**ANS. (Check if all rows have same elements or not)**

**m,n = int(input()),int(input())**

**prev = []**

**for i in range(m):**

**row = set(map(int,input().split()))**

**if prev != row and i != 0: print("Not Shuffled Row Matrix"); exit()**

**prev = row**

**print("Shuffled Row Matrix")**

**C++**

**#include <iostream>**

**#include <set>**

**using namespace std;**

**int main(){**

**int m,n;**

**cin>>m>>n;**

**set <int> prev;**

**for(int i=0;i<m;i++){**

**set <int> row;**

**for(int j=0;j<n;j++){**

**int temp; cin>>temp;**

**row.insert(temp);**

**}**

**//here we don't need any condition for i > 0 because this for loop doesn't start for prev**

**//as prev.begin() and prev.end() are same as prev is empty so unequal condition inside isn't**

**//tested, which results in the flow of code moving on thus, ignoring this part**

**for(auto x = row.begin(),y = prev.begin(); x != row.end(),y != prev.end(); x++,y++){**

**if(\*x != \*y){**

**cout<<"Not Shuffled Row Matrix";**

**exit(0);**

**}**

**}**

**prev.clear();**

**for(auto x: row) prev.insert(x);**

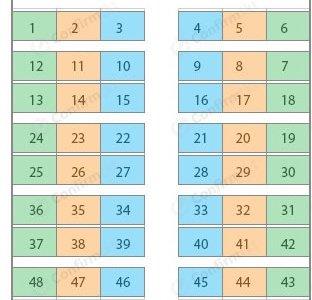
**}**

**cout<<"Shuffled Row Matrix";**

**}**

#### **Find Color of Chair**

In a game the participants are made to sit on chairs of three colors Green, Orange, Blue in a big hall. A number of pair of six chairs are arranged opposite to each other and numbered as shown in the following image. In the image each pair of six chairs are separated by a gap that is chairs with number 1 to 12 is a pair, 13 to 24 is a pair and so on.

****

The first chair is Green, second chair is Orange third chair is Blue in color, fourth is Blue, fifth is Orange and sixth is Green and then the colors of chairs are reversed so that color of chairs on opposite sides are same. The numbering of chairs are increasing from left to right in the first pair and is decreasing from left to right in the second pair. In this game a participant is called and given a chair number, ‘x’. He has to find the number of the chair opposite to ‘x’ and also find the color of ‘x’. For example, when ‘x’ is 30, print 31 and Green when ‘x’ is 46 print 39 and Blue.

**Input Format**

First line contains the chair number, ‘x’

**Output Format**

First line contain the chair ‘y’ number opposite to ‘x’

Next line contain the color of chair ‘y’ (Green or Orange or Blue)

**ANS.**

**nums=[]; l1=[]**

**for i in range(1,109): nums.append(i)**

**for i in range(0,len(nums),6):**

**l=[]**

**for j in range(i,i+6):**

**l.append(nums[j])**

**l1.append(l)**

**for i in range(len(l1)):**

**if i%2!=0:**

**l1[i].reverse()**

**all\_pairs=[]**

**for i in range(0,len(l1),2):**

**pair=tuple(zip(l1[i],l1[i+1]))**

**for j in pair:**

**all\_pairs.append(j)**

**all\_pairs.append(tuple(reversed(list(j))))**

**pairs\_for\_ans=dict(sorted(all\_pairs))**

**colours=["Green","Orange","Blue","Blue","Orange","Green"]**

**colour\_code=[]**

**for i in l1:**

**colour\_pair=tuple(zip(i,colours))**

**for j in colour\_pair:**

**colour\_code.append(j)**

**colours\_for\_ans=dict(sorted(colour\_code))**

**n=int(input())**

**x=pairs\_for\_ans[n]**

**print(x)**

**print(colours\_for\_ans[x])**

#### **Help to Shift Library**

In a village, the local library is to be shifted from one place to the other. Books are to be taken from the old location to the new one, there are no vehicles to help the librarian. Two children Ramu and Somu volunteered to help the librarian in the shifting process. Since there are many books shifting will require many trips. The librarian thinks that the children will become tired over time. So he plans to shift the bigger books first and then the smaller books. To make the shifting process interesting he asks Ramu to carry ‘i’ books and Somu to carry ‘2\*i’ books in the odd trips and during even trips Ramu has to carry ‘2\*i’ books and Somu has to carry ‘i’ books. Where ‘i’ is the trip number and the trips are numbered from 1.

Given the total number of books in the library, write a C program to find out the number of trips required to shift the books, number of books carried by Ramu, number of books carried by Somu and the name of the child that keep the last book in the new location of the library.

For example, when there are 27 books, it will take totally four trips to complete the shifting, Ramu would carry 16 books, Somu would carry 11 books, and the last book will be placed by Somu. When there are 48 books it will take totally six trips to shift, Ramu would carry 24 books, Somu would carry 24 books, and the last book will be placed by Ramu.

**Input Format**

First line contains the total number of books in the library, n

**Output Format**

First line should contain the number of trips taken for shifting

Nex line should contain the number of books carried by Ramu and Somu separated by a tab

Next line should contain the name of the child who carried the last book

**ANS.**

**n=int(input()); ramu=[]; somu=[]; i=1; flag=0**

**while n!=0: #both cases have Ramu getting his books first for carrying**

**if i%2==0:**

**if n>2\*i: ramu.append(2\*i); n-=2\*i**

**else: ramu.append(n); n-=n; flag=1**

**if n>i: somu.append(i); n-=i**

**else: somu.append(n); n-=n**

**if i%2!=0:**

**if n>i: ramu.append(i); n-=i**

**else: ramu.append(n); n-=n; flag=1**

**if n>2\*i: somu.append(2\*i); n-=2\*i**

**else: somu.append(n); n-=n**

**i+=1**

**print(i-1)**

**print(sum(ramu),sum(somu),sep="\t")**

**print("Ramu" if flag==1 else "Somu")**

#### **Toggle Switch**

There are ‘n’ light bulbs lined up in a row in a long room. Each bulb is numbered consecutively from 1 to ‘n’ and each bulb is independently connected to a switch which are currently in off state. ‘n’ people are lined up outside the entry door of the room. Every ‘ith’ person in the line enter the room and toggle the state of ‘ith’ switch and exit the room. Given the value of ‘n’ and an integer ‘m’, write a C program to find the state of ‘mth’ switch. Print ‘On’ if the switch will be in ‘On’ state print ‘Off’ otherwise. For example, if there are 10 switches then status of 5th switch will be Off after all ten people toggle the switches.

**Input Format**

First line contains the value of ‘n’

Next line contains the value of ‘m’

**Output Format**

Print the status of the ‘mth’ switch

**ANS.**

**#we are checking no. of state changes at a single bulb about which we are asked**

**#if the flag%2 == 0 then both states have passed through it and it is on**

**#we started from off that is why both states completion means bulb is finally on**

**#by similar logic it would be off when flag%2 == 1**

**n = int(input())**

**m = int(input())**

**flag = 0**

**for i in range(1,m+1):**

**if m%i==0: flag+=1**

**if flag%2: print("On")**

**else: print("Off")**

#### **Secret Agent and Code Word**

A secret agent is a government employee whose job involves gathering secret information about the governments of unfriendly foreign countries. A secret agent ‘X’ emailed a sentence ‘S’ with a code word ‘W’ to his head office. Only one word of the sentence 'S' is real and others are fake. The agent ‘X’ also mailed a sentence as a clue - if I tell you any one character ‘C’ of the code word 'W', then you would find exactly two words with ‘C’ in it, those words will have same number of vowels and one of them is ‘W’.

For example, if the senetence ‘S’ mailed by agent ‘X’ is "AIM DUE OAT TIE MOD", then TIE is the code word. Due to the following reasons:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Word ‘W’** | **Letter in word (C)** | **Words with ‘C’ in it** | **Number of Vowels in the words** | **Status**  **(Consider/Reject) of pair of words** | **Status**  **(Consider/Reject) of word ‘W’** |
| AIM | A | AIM, OAT | AIM – 2  OAT – 2 | Consider |  |
| AIM | I | AIM, TIE | AIM – 2  OAT – 2 | Consider |  |
| AIM | M | AIM, MOD | AIM – 2  MOD – 1 | Reject | Reject |
| DUE | D | MOD, DUE | DUE – 2  MOD – 1 | Reject | Reject |
| OAT | O | OAT, MOD | OAT – 2  MOD – 1 | Reject | Reject |
| MOD | M | MOD, AIM | MOD – 1  AIM – 2 | Reject | Reject |
| TIE | T | OAT, TIE | OAT - 2  TIE – 2 | Consider |  |
| TIE | I | TIE, AIM | TIE – 2  AIM – 2 | Consider |  |
| TIE | E | TIE, DUE | TIE – 2  DUE – 2 | Consider | Consider |

**Input Format**

First line contains the sentence with codeword

**Output Format**

First line should contain the code word

**Boundary Conditions**

Length of the sentence < 200

Length of each word in the sentence<10

Number of words in each sentence <20

**ANS.**

**def vowel(s):**

**vowels="AEIOU"; count=0**

**for i in s:**

**if i in vowels:**

**count+=1**

**return count**

**def any\_letter\_common(s,s1):**

**flag=0**

**for i in s:**

**if i in s1:**

**flag=1**

**return i**

**if flag==0: return flag**

**def all\_letters\_checker(s,l):**

**flag=1**

**for i in s:**

**if i not in l:**

**flag=0**

**return flag**

**if flag==1: return flag**

**s=input().split()**

**possible\_words=set(); possible\_letters=set()**

**for i in range(len(s)):**

**for j in range(i+1,len(s)):**

**if vowel(s[i])==vowel(s[j]) and any\_letter\_common(s[i],s[j])!=0:**

**possible\_letters.add(any\_letter\_common(s[i],s[j]))**

**possible\_words.add(s[i]); possible\_words.add(s[j])**

**for i in possible\_words:**

**if all\_letters\_checker(i,possible\_letters)==1: print(i)**

**ANOTHER APPROACH (MORE LIKE THE QUESTION ASKED):**

**from collections import Counter**

**s = input().split()**

**letter\_freq = Counter("".join(s))**

**#for getting the possible letters that can be common between any two words**

**possible\_letters = [i for i in letter\_freq if letter\_freq[i] > 1]**

**def vowel\_count(w): #for counting no. of vowels**

**count = 0**

**for i in w:**

**if i in "AEIOU": count+=1**

**return count**

**def check(w1,w2): #for checking conditions given for two words that are**

**#2 vowels in each and 1 common letter**

**if vowel\_count(w1) == vowel\_count(w2) == 2:**

**common\_letters = set(list(w1)).intersection(set(list(w2)))**

**for i in common\_letters:**

**if i in possible\_letters:**

**return True**

**return False**

**freq = dict(zip(s,[0]\*len(s))) #for keeping count of how many times any word**

**#satisfies the conditions given with one of the other words**

**for i in range(len(s)-1):**

**for j in range(i+1,len(s)):**

**if check(s[i],s[j]):**

**freq[s[i]]+=1**

**freq[s[j]]+=1**

**value\_max = max(freq.values())**

**for i in freq:**

**if freq[i] == value\_max:**

**print(i); exit()**

**C++:**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**bool check(string w1,string w2){**

**for(int i=0;i<w1.length();i++){**

**char key = w1[i];**

**for(int j=0;j<w2.length();j++){**

**if(w2[j] == key) return true;**

**}**

**}**

**return false;**

**}**

**int vowel\_count(string s){**

**int cnt = 0;**

**char vowels[5] = {'A','E','I','O','U'};**

**for(int i=0;i<s.length();i++){**

**for(int j=0;j<5;j++){**

**if(s[i] == vowels[j]) cnt++; //A break statement would have made this more efficient**

**}**

**}**

**return cnt;**

**}**

**int main(){**

**string sentence;**

**getline(cin,sentence); //way to input strings with spaces in C++**

**vector <string> words; // way to separate words from a space-containing sentence**

**string word = "";**

**int cnt = 0;**

**for(int i=0;i<sentence.length()+1;i++){**

**if(sentence[i] == ' ' || sentence[i] == '\0'){**

**cnt++; words.push\_back(word); word = "";**

**}**

**else word+=sentence[i];**

**}**

**vector <int> freq(cnt,0);**

**for(int i=0;i<cnt-1;i++){**

**for(int j=i+1;j<cnt;j++){**

**int x = vowel\_count(words[i]),y = vowel\_count(words[j]);**

**if(check(words[i],words[j]) && x==y && x==2){**

**freq[i]++; freq[j]++;**

**}**

**}**

**}**

**int max\_freq = 0;**

**for(int i=0;i<cnt;i++){**

**if(freq[i] > max\_freq) max\_freq = freq[i];**

**}**

**for(int i=0;i<cnt;i++){**

**if(freq[i] == max\_freq){**

**cout<<words[i];**

**exit(0);**

**}**

**}**

**}**

#### **Ways to Place Pot of Milk on Stools**

‘n’ girls have arrived with pots of milk and these pots of milk has to arranged on ‘n’ stools which are arranged in a single row in a hall. The girls has are made to stand in a queue as per their order of arrival and place the pot on any one of the stool in the hall satisfying the rules:

1. First girl can keep the pot on whichever stool she wants

2. ‘nth’ girl where n>1, can keep the pot on a stool which is adjacent to a non-empty stool

For example, when there is only one stool and only one girl ’x’ in the queue, there is only one option to place the milk pot

****

When there are two stools and there are two girls ‘x’ and ‘y’ in the queue, there are two ways to arrange the milk pots carried by them either ‘x’ -> ‘y’ or ‘y’->’x’

****

When there are three girls ‘x’, ‘y’ and ‘z’ in the queue, there are four ways to arrange the milk pots carried by them. ‘x’->’y’->’z’, ‘z’->’y’->’x’, ‘y’->’x’->’z’, ‘z’ -> ‘y’ ->’x’

****

Given the value of n’, where ‘n’ is the number of girls with milk pot and the number of stools in the hall, write a C program to find the number of ways in which the milk pots can be arranged. For example, if n is 3 then there are 4 ways as illustrated above.

**Input Format**

First line contains the value of ‘n’

**Output Format**

Print the number of ways in which the pot can be arranged

**ANS. print(2\*\*(int(input())-1))**

#### **Arithmetic of Secret Agent**

Another secret agent ‘Y’ communicates the secret word to his head office as two words ‘W1’ and ‘W2’. The number of characters in the codeword is one more than the maximum of length of ‘W1’ and ‘W2’. He has given instruction to his head office to do arithmetic on the letters of the alphabet using the following rules:

1. Addition happens similar to numeric addition and starts from the right most character of the words.
2. The ASCII value of the characters is used for addition. The right most digit of the result is taken as sum and the remaining digits are considered as carry. The letter corresponding to the character position in English alphabet of sum will be the character in the codeword (i.e.), 0-a, 1-b, 2-c, …, 25-z.
3. The carry is added along with the next character to the left of the current character.
4. The steps 1-3 are repeated till the left most character of the of the longest word among ‘W1’ and ‘W2’ is reached. When end of the words ‘W1’ or ‘W2’ is reached then the remaining characters of the longer word alone take part in addition with carry.
5. The last sum got will be the next to the leftmost character of the codeword. The carry left out will give the leftmost character of codeword and it may have more than one digit. If the value exceeds 25, then the equivalent character in the alphabet is obtained in a cyclic manner. ie., 26-a, 27-b and so on.

Given two words W1 and W2, develop a pseudocode and a C code to find the codeword sent by the agent.

For example, if S1= ‘zap’ and S2= ‘zap’, then the numerical is sum 26564 and the equivalent alpha sum is obtained as follows:

26-‘a’, 5- ‘f’, 6-‘g’, 4=’e’.

Hence, the alpha sum is ‘afge’

**ANS.**

**alpha\_code = {}**

**for i in range(0,26):**

**alpha\_code[i] = chr(97+i)**

**w1,w2 = input(),input()**

**major,minor = "",""**

**if len(w1) >= len(w2): minor = w2; major = w1**

**else: minor = w1; major = w2**

**ans = []**

**carry = 0**

**for i in range(1,len(major)+1):**

**if i <= len(minor):**

**d = ord(major[-i])+ord(minor[-i])+carry**

**else:**

**d = ord(major[-i])+carry**

**carry = (d-d%10)//10**

**if i < len(major): ans.append(d%10)**

**else: ans.append(d%10); ans.append(carry)**

**codeword = ""**

**for i in range(len(ans)-1,-1,-1):**

**codeword+=alpha\_code[ans[i]%26]**

**print(codeword)**

#### **Count Beeps**

Binary representation of a number ‘n’ are written on a circle which is divided into ‘m’ segments.

Here the number of bits required to represent ‘n’ is ‘m’.

A needle is fixed on the top of this circle and the circle rotates in clockwise direction for every ten seconds

till a complete rotation is made that is the number of rotations is equal to ‘m-1’.

Let the segment pointed by needle be ‘p’. A beep sound is made when the decimal equivalent of the

binary digits is even when read from the next segment to the right of ‘p’ to the segment ‘p’.

Given the binary digits written on the segment from the segment next to ‘p’ to ‘p’,

write a C program to find the number of times beep sound will be made. For example, if the initial

configuration of binary string given is ‘1001101’ then the binary strings obtained by rotation are

1001101

1100110

0110011

1011001

1101100

0110110

0011011

The binary strings 1100110, 1101100 and 0110110 are even when decimal equivalent is taken.

Hence beep sound will be made three times.

**Boundary Conditions**

Length of the binary string < 25

**Input Format**

First line contains the binary representation of the number ‘n’

**Output Format**

Print the number of times beep sound will be made

**ANS.**

**from collections import \***

**n=deque(input()); beep=0**

**for i in range(len(n)):**

**n.rotate(-1); x=int("".join(list(n)),2)**

**if x%2==0: beep+=1**

**print(beep)**

**ALTERNATE SOLN: print(input().count(“0”))**

#### **Classify Triplets**

Given two integers ‘l’ and ‘u’, write a C program to identify the triplet (a, b, c) where a <= b <= c and a, b, c >= l and a, b, c <= u, that are Exactly Pythoagorean, Very Close to Pythagorean or Close to Pythagorean. A triplet (a, b, c) is said to be Exactly Pythagorean if c2 – a2+b2 = 0, Very Close to Pythagorean if c2 – a2+b2 = +1 or -1 and Close to Pythagorean if c2 – a2+b2 = +2 or -2. For example, if l = 5 and u = 20 then the program should print

5 5 7 Very Close to Pythagorean

5 11 12 Close to Pythagorean

5 12 13 Exactly Pythagorean

5 13 14 Close to Pythagorean

6 8 10 Exactly Pythagorean

6 17 18 Very Close to Pythagorean

6 18 19 Very Close to Pythagorean

7 7 10 Close to Pythagorean

7 11 13 Very Close to Pythagorean

8 9 12 Very Close to Pythagorean

8 15 17 Exactly Pythagorean

9 12 15 Exactly Pythagorean

10 15 18 Very Close to Pythagorean

11 13 17 0Very Close to Pythagorean

12 12 17 Very Close to Pythagorean

12 16 20 Exactly Pythagorean

**Input Format**

First line contains the value of ‘l’

Next line contains the value of ‘u’

**Output Format**

Print all the triplets that are Exactly, Very Close to or Close to Pythagorean triplets

**ANS.**

**l=int(input());u=int(input())**

**for i in range(l,u+1):**

**for j in range(l,u+1):**

**for k in range(l,u+1):**

**if (l<=i<=j<=k<=u) and (i and j and k >=1):**

**if i\*\*2+j\*\*2==k\*\*2: print(i,j,k,"Exactly Pythagorean")**

**if i\*\*2+j\*\*2==k\*\*2+1 or i\*\*2+j\*\*2==k\*\*2-1: print(i,j,k,"Very Close to Pythagorean")**

**if i\*\*2+j\*\*2==k\*\*2+2 or i\*\*2+j\*\*2==k\*\*2-2: print(i,j,k,"Close to Pythagorean")**

#### **Hands of the Clock**

In the dial of an analog clock, tweleve alternate letters of the English alphabet are written to indicate 1 to 12.

****

Capital letters are written and it may start from any letter A to D. Given the letter from which numbering starts, a hour of a time, a minute of a time, write a C program to find out the position of hour hand and minute hand. For example, if the numbering starts from ‘D’ and the hour of the given time is 19 and minute of the given time is 45 then the hour hand will be “Between P and R” and minute hand will be “On T”. If the time is 07:00 then the hour hand will be “On P” and minute hand will be “On Z”. Only valid time will be given as input, hour will be given from 0 to 24 and minute will be from 0 to 59.

**Input Format**

First line contains the letter from which the numbering starts

Next line contains the hour of the time

Next line contains minute of the time

**Output Format**

In the first line, print the position of hour hand – Print On a particular letter or Between two letters

In the next line, print the position of minute hand

**ANS.**

**from math import ceil**

**# input line**

**s=input().strip(); h=int(input())%12; m=int(input())/5**

**# dictionary to store numbers and their alphabetical values**

**code = {}; count=0**

**for i in range (0,24,2): count+=1; code[count]=chr(ord(s)+i)**

**# when dealing with times around the noon and midnight in 24 hr format**

**if h==0:**

**if m==0:**

**print("On",code[12])**

**print("On",code[12]); exit()**

**if 0<m<1:**

**print("Between",code[12],"and",code[1])**

**print("Between",code[12],"and",code[1]); exit()**

**if 1<=m<12:**

**print("Between",code[12],"and",code[1])**

**if int(m)==m: print("On",code[m]); exit()**

**else: print("Between",code[int(m)],"and",code[ceil(m)]); exit()**

**# when dealing with times between the noon and midnight**

**if h<12:**

**if m==0:**

**print("On",code[h])**

**print("On",code[12]); exit()**

**if 0<m<1:**

**print("Between",code[h],"and",code[h+1])**

**print("Between",code[12],"and",code[1]); exit()**

**if 1<=m<12:**

**print("Between",code[h],"and",code[h+1])**

**if int(m)==m: print("On",code[m]); exit()**

**else: print("Between",code[int(m)],"and",code[ceil(m)]); exit()**

#### **Count Same Positions**

Given an integer ‘n’, write a C program to count the number of digits that are in the same position after forming an integer ‘m’ with the digits in ‘n’ but in ascending order of digits. For example, if the value of ‘n’ is 351462987 then value of ‘m’ will be 123456789 and digits 4 and 8 will be in the same position.

**Input Format**

First line cotains the value of ‘n’

**Output Format**

Print the position of digits (ascending order) in ‘n’ that has same value in ‘m’ also. Print one position in one line

**Boundary Conditions**

0 < n <1019

**ANS.**

**n=list(str(int(input()))); p=sorted(n)**

**for i in range(len(p)):**

**if n[i]==p[i]: print(i+1)**

**C CODE USING STRINGS:**

**#include <stdio.h>**

**#include <string.h>**

**char\* sorter(char arr[]){**

**for (int i=0;i<strlen(arr);i++){**

**for(int j=i+1;j<strlen(arr);j++){**

**int temp;**

**if (arr[i]>arr[j]){**

**temp=arr[i];**

**arr[i]=arr[j];**

**arr[j]=temp;**

**}**

**}**

**}**

**return(arr);}**

**int main(){**

**char num[100],num\_original[100];**

**scanf("%[^\n]\*c",&num);**

**strcpy(num\_original,num);**

**char\* sorted\_num=sorter(num);**

**for (int i=0;i<strlen(sorted\_num);i++){if (num\_original[i]==sorted\_num[i]){printf("%d\n",i+1);}}**

**}**

**C PROGRAM USING SELECTION SORT:**

**#include <stdio.h>**

**#include <string.h>**

**//JO BHI DIFFERENT HAI BUBBLE SORT SE YELLOW MAIN**

**char\* sorter(char arr[]){**

**for (int i=0;i<strlen(arr);i++){**

**for(int j=0;j<strlen(arr)-i-1;j++){**

**int temp;**

**if (arr[j]>arr[j+1]){**

**temp=arr[j];**

**arr[j]=arr[j+1];**

**arr[j+1]=temp;**

**}**

**}**

**}**

**return(arr);**

**}**

**int main(){**

**char num[100],num\_original[100];**

**scanf("%[^\n]\*c",&num);**

**strcpy(num\_original,num);**

**char\* sorted\_num=sorter(num);**

**for (int i=0;i<strlen(sorted\_num);i++){if (num\_original[i]==sorted\_num[i]){printf("%d\n",i+1);}}**

**}**

**NUMBERS KO STRINGS JAISE ACCESS KRNA HO TOH AISE KRTE C MAIN AUR ARRAYS RETURN KAISE KARATE WHO BHI DEKH LO**

#### **Code Word Check**

An agent ‘A’ comes to a secret place to collect some information, the information will be disclosed to him if he is able to find one of the code words of the group of agents from a given word ‘w’. All the code words are of length four. The agent ‘A’ will say the positions of four letters in ‘w’ that may form the code word. Write a C program to check if the letters in ‘w’ in the positions mentioned by ‘A’ is a code word. For example, if the valid code words are HEAR, BEAR, VEST, MONK and if ‘w’ is ‘RESEARCH’ and position mentioned by the agent is 8, 2, 5 and 6 then it is Valid. Position of letters in the word ‘w’ starts from 1. Print Valid if the word that can be formed by the positions mentioed by the agent ‘A’ is in the list of code words and print Invalid otherwise.

**Input Format**

First line contains the number of code words among the group of agents, n

Next ‘n’ lines contain the code words

Next line contains the word, ‘w’

Next four lines contain the position of the letters chosen by the agent ‘A’

**Output Format**

Print either Valid or Invalid

**ANS.**

**n=int(input())**

**words=[input() for i in range(n)]**

**w=input(); x=""**

**pos=[int(input()) for i in range(4)]**

**for i in pos: x+=w[i-1]**

**if x in words: print("Valid")**

**else: print("Invalid")**

**C CODE:**

**#include <stdio.h>**

**#include <string.h>**

**// jitna chaie usse ek zyaada space leke chalo arrays mein**

**int main(){**

**int n;**

**scanf("%d",&n);**

**char words[n+1][5];**

**char code\_word[20];**

**int pos[5];**

**char test\_word[5];**

**for (int i=0;i<n;i++){scanf("%s",&words[i]);}**

**scanf("%s",&code\_word);**

**for(int i=0;i<4;i++){scanf("%d",&pos[i]);}**

**for(int i=0,j=0;i<4;i++){test\_word[j++]=code\_word[pos[i]-1];}**

**bool flag=0;**

**for (int i=0;i<n;i++){if (strcmp(test\_word,words[i])==0){ printf("Valid"); return 0;}}**

**printf("Invalid");**

**}**

#### **Sides of a Rectangle**

Given four points P1, P2, P3 and P4 check whether they can be corners of a rectangle with sides A, B, C and D respectively. The naming of the sides of the rectangle are shown as below.

****

For example, if P1 is (2, 12), P2 is (14, 12), P3 is (14, 22), P4 is (2, 22) then they can be corners A, B, C and D of a rectangle. Whereas the points P1 is (5, 12), P2 is (14, 12), P3 is (14, 22), P4 is (2, 22) cannot be corners of a rectangle. Distance between any two points P1(x1, y1) and P2(x2, y2) shall be calculated using the formula:

****

Print Yes if the points P1, P2, P3 and P4 can be corners A, B, C and D of a rectangle respectively and No otherwise.

**Input Format**

First line contains the coordinates of point P1 separated by a space

Next line contains the coordinates of point P2 separated by a space

Next line contains the coordinates of point P3 separated by a space

Next line contains the coordinates of point P4 separated by a space

**Output Format**

Print Yes or No

**ANS.**

**# all opposite sides equal to each other and both diagonals are equal to each other**

**def distance(x1,y1,x2,y2):**

**dis = ((x2-x1)\*\*2+(y2-y1)\*\*2)\*\*0.5; return dis**

**x1,y1 = map(int,input().split())**

**x2,y2 = map(int,input().split())**

**x3,y3 = map(int,input().split())**

**x4,y4 = map(int,input().split())**

**a = distance(x1,y1,x2,y2)**

**b = distance(x2,y2,x3,y3)**

**c = distance(x3,y3,x4,y4)**

**d = distance(x4,y4,x1,y1)**

**dg1 = distance(x3,y3,x1,y1)**

**dg2 = distance(x4,y4,x2,y2)**

**if a == c and b == d and dg1 == dg2: print("Yes")**

**else: print("No")**

#### **Find the Week Day**

Given the week day of nth day of a month, write a C program to find mth day of the month. For example, if it is given that 16th of the month is Wednesday then 25th of the month is Friday and if it is given that 29th of the month is Tuesday then 2nd of the month is Wednesday. Weekday number is given as input for ‘n’ that 0 indicates Sunday, 1 indicates Monday and so on.

**Input Format**

First line contains the value of n

Next line contains the week day number of ‘n’

Next line contains the value of ‘m’

**Output Format**

Print the week day of m. First letter of weekday should be in upper case

**ANS.**

**week={1:"Monday",2:"Tuesday",3:"Wednesday",4:"Thursday",5:"Friday",6:"Saturday",7:"Sunday"}**

**date=int(input())**

**day\_no=int(input())**

**find=int(input())**

**if find>date:**

**p=find-date**

**x=day\_no+p%7**

**if x>7: print(week[x%7])**

**else: print(week[x])**

**elif find<date:**

**p=date-find**

**x=day\_no-p%7**

**if x>0: print(week[x])**

**else: print(week[7+x])**

**else: print(week[day\_no])**

**ALTERNATE SOLUTION:**

**dow = {0:"Sunday",1:"Monday",2:"Tuesday",3:"Wednesday",4:"Thursday",5:"Friday",6:"Saturday"}**

**n = int(input())**

**day = int(input())**

**m = int(input())**

**if m > n:**

**print(dow[(day+(m-n)%7)%7])**

**elif m < n:**

**x = day-(n-m)%7**

**if x >= 0: print(dow[x])**

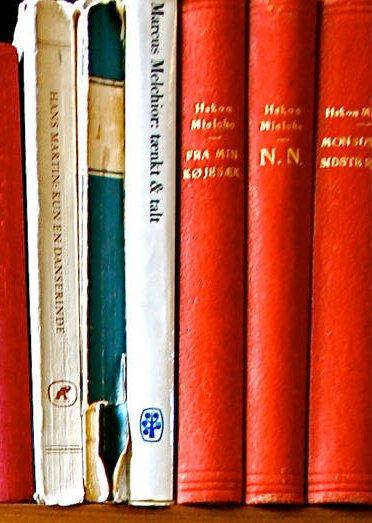
**else: print(dow[x+7])**

**else:**

**print(dow[day])**

#### **Distance Travelled by Ant**

Books in a library shelf are arranged as shown below and an ant travels on the top of the books from front cover of nth book to the back cover of mth book. The ant may travel in either direction (left or right) of the shelf. Assume that all the books in the shelf are of uniform thickness and thickness of the covers are negligible.

****

Given the direction in which the ant moves, the value of ‘m’,‘n’ and thickness of one book, write a C program to find the distance travelled by the ant. For example, when the ant travels from 7th book to the 12th book from left side to right side of the shelf, thickness of one book is 3 cm then the distance travelled by the ant is 12cm and when the ant travels from 5th book to 13th book from the right side of the shelf to the left side of the shelf and thickness of one book is 2.5 cm then the ant will travel 22.50 cm. Print two decimal places of the distance travelled by the ant.

**Input Format**

First line contains the direction of movement of ant, ‘l’ indicates that the ant moves from left to right side of the shelf and ‘r’ indicates that the ant moves from right to left side of the shelf

Next line contains the value of ‘n’

Next line contains the value of ‘m’

Next line contains the thickness of the book

**Output Format**

Print the distance travelled by the ant, d. Print only two decimal places for distance

**ANS.**

**dir,n,m,t=input(),int(input()),int(input()),float(input())**

**print(format((m-n-1)\*t,'0.2f') if dir=="l" else format((m-n+1)\*t,'0.2f'))**

**Alternate:**

**d = input()**

**n,m,ans = int(input()),int(input()),0**

**t = float(input())**

**if d == 'l': ans = (m-n-1)\*t**

**else: ans = (m-n+1)\*t**

**print('{:.2f}'.format(ans))**

**C code:**

**#include <stdio.h>**

**//remember the ' ' quotes and " " quotes are different in c and c++**

**int main(){**

**int m,n;**

**float t;**

**char dir;**

**scanf("%c\n%d\n%d\n%f",&dir,&n,&m,&t);**

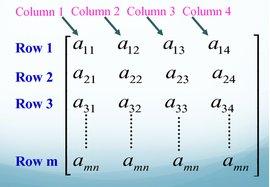
**if (dir=='l'){printf("%0.2f",(m-n-1)\*t);}**

**else {printf("%0.2f",(m-n+1)\*t);}**

**}**

#### **Maximum Z-Sum**

Z-Sum(i,j) in a matrix M is defined as the sum of the elements in matrix M at positions (i, j), (i, j+1), (i+1, j), (i+1, j+1). Z-Sum(1,1) of the matrix shown in the figure is a11+a12+a21+a22.

****

Given a mXn matrix, value of i and j, write a C program to find the maximum Z-Sum in the matrix. Print the elements that form Z-structure in order that is print elementsat positions (i, j), (i, j+1), (i+1, j), (i+1, j+1). And also print all the positions i,j of the matrix that corressponds to maximum Z-Sum. For example, given a 3X3 matrix as shown below:

1 2 3

2 4 7

1 2 3

The maximum z-sum is 16, the z-structures start with elements in position 1, 2 and 2, 2, the elements that is on the maximum z-structures are 2 3 4 7 and 4 7 2 3.

**Input Format**

First line contains the number of rows in the matrix m, r

Next line contains the number of columns in the matrix m, c

Next r lines contain the elements of the matrix, one line has the elements of one row and the elements in each column is separated by a space

**Output Format**

In the first line, print the maximum z sum

In the next 2\*m lines, print the position of first element of Z-structure that has maximum z sum followed the elements in the Z-structure with maximum sum in the next line, where ‘m’ is the number of z-structures with maximum z sum

The position of first element of Z-structure with maximum sum and elements in the Z-structure with maximum sum has to be separated by a tab space. Print the elements in the Z-structure in such a way that Z shape is formed when traversed

**ANS.**

**PYTHON CODE:**

**r,c=int(input()),int(input())**

**m=[list(map(int,input().split())) for i in range(r)]**

**sums=[];info=[]**

**for i in range(r-1):**

**for j in range(c-1):**

**x=m[i][j]+m[i][j+1]+m[i+1][j+1]+m[i+1][j]**

**v=(m[i][j],m[i][j+1],m[i+1][j],m[i+1][j+1])**

**info.append((x,i+1,j+1,v))**

**sums.append(x)**

**print(max(sums))**

**for i in info:**

**if i[0]==max(sums):**

**print(i[1],i[2],sep="\t")**

**for k in i[3]:**

**if i[3].index(k)!=len(i[3])-1: print(k,end="\t")**

**else: print(k)**

**C CODE:**

**#include <stdio.h>**

**int main(){**

**int r,c;**

**scanf("%d\n%d",&r,&c);**

**int m[r][c],max=NULL;**

**for(int i=0;i<r;i++){for(int j=0;j<c;j++){scanf("%d",&m[i][j]);}}**

**for(int i=0;i<r-1;i++){**

**for (int j=0;j<c-1;j++){**

**int sum = m[i][j]+m[i+1][j]+m[i][j+1]+m[i+1][j+1];**

**if (sum>max){max=sum;}**

**}**

**}**

**printf("%d\n",max);**

**for(int k=0;k<r-1;k++){**

**for(int l=0;l<c-1;l++){**

**if (m[k][l]+m[k+1][l]+m[k][l+1]+m[k+1][l+1]==max){**

**printf("%d\t%d\n",k+1,l+1);**

**printf("%d\t%d\t%d\t%d\n",m[k][l],m[k][l+1],m[k+1][l],m[k+1][l+1]);**

**}**

**}**

**}}**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**int r,c;**

**cin>>r>>c;**

**//input for matrix m**

**int m[r][c];**

**for(int i=0;i<r;i++){**

**for(int j=0;j<c;j++){**

**cin>>m[i][j];**

**}**

**}**

**vector<int> zsums;**

**vector<pair<int,int>> coors;**

**//calculating all z-sums**

**for(int i=0;i<r-1;i++){**

**for(int j=0;j<c-1;j++){**

**zsums.push\_back(m[i][j]+m[i][j+1]+m[i+1][j]+m[i+1][j+1]);**

**coors.push\_back({i,j});**

**}**

**}**

**//max of all z-sums**

**int max\_zsum = \*max\_element(zsums.begin(),zsums.end());**

**cout<<max\_zsum<<"\n";**

**//printing the max z-sums' coordinates**

**for(int i=0;i<coors.size();i++){**

**if(zsums[i] == max\_zsum){**

**int k = coors[i].first,l = coors[i].second;**

**cout<<k+1<<"\t"<<l+1<<"\n";**

**cout<<m[k][l]<<"\t"<<m[k][l+1]<<"\t"<<m[k+1][l]<<"\t"<<m[k+1][l+1]<<"\n";**

**}**

**}**

**}**

**Alpha garland**

Given an alphanumeric string S, develop pseudocode and C code for printing the alpha garland by processing S character by character based on the following rules:

If the current character is an alphabet, add it to the garland.

If the current character is a number, then the garland is extended by adding the alphabets present so far in the garland in reverse form and then its original form alternatively.

The process is repeated till the end of the string.

For example, if the S= “ab3c1” ,then the garland is “ab baabba c cabbaabba”

***Note:*** The white space in the garland is given for understanding the process. Actually the garland formed for this string is “abbaabbaccabbaabba”

**Input Format**

An alphanumeric string, S

**Output Format**: A string forming the alpha garland

**ANS.**

**ip=input()**

**op=""**

**for i in range(len(ip)):**

**if 96<ord(ip[i])<123: op+=ip[i]; x=op**

**else:**

**y="".join(list(reversed(list(op))))**

**for j in range(int(ip[i])):**

**if j%2==0: op+=y**

**else: op+=x**

**print(op)**

**NEW CODE:**

**s,ans = input(),""**

**for i in range(len(s)):**

**x = ord(s[i])-48**

**if x in {1,2,3,4,5,6,7,8,9}:**

**addition = ans**

**for i in range(x):**

**if i%2: ans+=addition**

**else: ans+=addition[::-1]**

**else:**

**ans+=s[i]**

**print(ans)**

**C++ CODE:**

**#include <iostream>**

**#include <cstring>**

**using namespace std;**

**string strrev(string s){**

**string ans = "";**

**for(int i=s.length()-1;i>-1;i--) ans+=s[i];**

**return ans;**

**}**

**int main(){**

**string s,ans="";**

**cin>>s;**

**for(int i=0;i<s.length();i++){**

**int x = s[i]-48;**

**string addition = ans;**

**if(x < 10){**

**for(int i=0;i<x;i++){**

**if(i%2) ans+=addition;**

**else ans+=strrev(addition);**

**}**

**}**

**else ans+=s[i];**

**}**

**cout<<ans;**

**}**

**Q. MATH CUM TIE SKILL COMPETITION:**

**You have to tie three strings of 20 beads and given a number n which you have to continuously divide by 3 to get remainder if for a particular time remainder is 0=red(r),1=green(g) and 2=blue(b) you have to put beads in the string like a string is being added with r,g,b chars. Then you have to find two numbers greater than n whose strings have exactly one and two beads different from the string we made for n. If the remainder becomes zero before filling the string completely then put “r” in place to fill the remaining spaces. String is filled from left to right basically reverse the string formed after your pattern making operations.**

**INPUT- n**

**OUTPUT-**

**Num1 Num2(separated by tabspace) (with one bead different before and two beads different after)**

**Pattern of n**

**Pattern of num1**

**Pattern of num2**

**def pattern(n):**

**s=''; rem=0; k=0**

**while k<20:**

**rem=n%3**

**if rem==0: s+='r'**

**if rem==1: s+='g'**

**if rem==2: s+='b'**

**n//=3**

**k+=1**

**return "".join(s[::-1])**

**def remove\_0(s):**

**for i in range(len(s)):**

**if len(s[:i])!=s[:i].count("r"): return s[i-1:]**

**n=int(input())**

**z=pattern(n)**

**k=n+1; x=""; num1=0**

**while True:**

**s=remove\_0(pattern(k))**

**comm=0**

**for i in range(len(s)):**

**if s[i]!=remove\_0(z)[i]: comm+=1**

**if comm==1: x=pattern(k); num1=k; break**

**k+=1**

**k1=n+1; y=""; num2=0**

**while True:**

**s1=remove\_0(pattern(k1))**

**comm1=0**

**for i in range(len(s1)):**

**if s1[i]!=remove\_0(z)[i]: comm1+=1**

**if comm1==2: y=pattern(k1); num2=k1; break**

**k1+=1**

**print(num1,num2,sep="\t")**

**print(pattern(n))**

**print(x)**

**print(y)**

**NEW CODE:**

**from collections import deque**

**def func(n):**

**s,cnt,f,atnzero = deque(),0,0,0**

**while(cnt < 20):**

**if n == 0 and f == 0: atnzero = cnt; f = 1**

**r = n%3**

**if r == 0: s.appendleft('r')**

**elif r == 1: s.appendleft('g')**

**else: s.appendleft('b')**

**n//=3**

**cnt+=1**

**return ("".join(s),atnzero)**

**def check(t1,t2):**

**g = max(t1[1],t2[1])**

**s1,s2 = t1[0][-g:],t2[0][-g:]**

**ucnt = 0**

**for i in range(g):**

**if s1[i] != s2[i]: ucnt+=1**

**return ucnt**

**n = int(input())**

**s = func(n)**

**num = n+1**

**f1,f2 = 0,0**

**n1,n2 = 0,0**

**s1,s2 = "",""**

**while(f1 == 0 or f2 == 0):**

**x = func(num)**

**if f1 == 0 and check(s,x) == 1:**

**n1 = num; s1 = x[0]; f1 = 1**

**if f2 == 0 and check(s,x) == 2:**

**n2 = num; s2 = x[0]; f2 = 1**

**num+=1**

**print(n1,n2,sep="\t")**

**print(s1,s2,sep="\n")**

#### **Distribution of Biscuit Packets**

A charity wants to help the people in ‘m’ number of villages of a district that were badly affected by recent November floods. They have got some funds with them with which they would be able to get a maximum ‘n’ number of biscuit packets. A rich man in the city has told that he will also join the charity in helping the people with the following conditions:

(i) He will give biscuit packets before going to each village. If the charity has ‘p’ number of biscuit packets before going to a village then he will give ‘p’ more biscuit packets.

(iii) The charity should give equal number of biscuit packets to all the villages

(iv) No biscuit packets should be left over after the distribution process

Given the number of villages ‘m’, the maximum number of biscuit packets ‘n’ that can be bought with the money in the hands of the charity, write a C program to find the number of biscuit packets that should be bought so that maximum number of biscuit packets can be distributed to the villages and the number of biscuit packets that shall be distributed to each village.

For example, if the number of villages is 2, and the maximum number of biscuit packets that shall be bought with the money in the hands of charity is 10 then the charity shall buy 9 packets and distribute 12 packets to each village.

**Input Format**

First line contains the number of villages, m

Next line contains the maximum number of biscuit packets that can be bought with money in the hands of agency, n

**Output Format**

First line should contain the number of biscuit packets to be purchased

Next line should contain the number of biscuit packets to be distributed

**ANS.**

**m,n = int(input()),int(input())**

**per\_village\_share = 0**

**max\_init\_purchase = 0**

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

**x = (2/(2\*\*m-1))\*i**

**if x == int(x) and max\_init\_purchase < i:**

**max\_init\_purchase = i**

**per\_village\_share = x\*(2\*\*(m-1))**

**print(max\_init\_purchase,int(per\_village\_share),sep="\n")**

**# x+x\*2+x\*4... = 2\*(some number less than n) (n is the max allowed purchase)**

**# x\*(2\*\*(m-1)) or x\*last term of 2's power series which sums to 2\*\*m-1 = per village share**

**# m = no. of villages**

**# we have to find max some number less than n for which x is an integer not float**

#### **Search Titles of Books**

Ajay’s favourite book has title ‘t’, he is interested in finding the title of books in his school library whose title is made up of only letters of ‘t’ and has all letters in ‘t’ equal to or more number of times as in ‘t’. For example, if the title of the Ajay’s favourite book is “rail safety” then the book titled “fairy tales” and “yes its a rail safe” are the books of his interest. If no such book exist then print “No such book”. Given the title of Ajay’s favourite book and the title of books in his school library, write a C++ program to print all titles that are of interest or “No such book”.

**Input Format**

First line contains the title of the Ajay’s favourite book

Next line contains number of books in the school library, n

Next ‘n’ lines contain the title of the books

**Output Format**

Print all title of books that are of interest if exists and print No such book otherwise

**Note:**

1. Ignore spaces for processing

2. To read a string with spaces in C++ use getline function of cin object. For example to read a string with spaces into a character array, char s[100], write cin.getline(s,100);

3. After reading an integer or float using normal cin, use cin.ignore() before using cin.getline function

**ANS.**

**s=input().lower()**

**n=int(input())**

**t=[input() for i in range(n)]**

**t1=[i.lower() for i in t]; l=[]**

**for i in t1:**

**if "".join(sorted(set(i)))=="".join(sorted(set(s))): l.append(t[t1.index(i)])**

**if len(l)==0: print("No such book")**

**else:**

**for i in l: print(i)**

#### **Are Balls Shuffled?**

An electromagnet is a type of magnet in which the magnetic field is produced by an electric current. The magnetic field disappears when the current is turned off. Difference between an electromagnet and a permanent magnet is that the strength of magnetic field can be quickly changed by controlling the amount of electric current in the winding.

In an experiment, ‘n’ iron balls are placed on ‘n’ electromagnets in a circular fashion as shown below

****

The electromagnets are connected among themselves and they are rotated at a constant speed ‘c’ rad/sec. In this experiment one letter of English alphabet or a number is written on each ball. The character written on the balls are unique. During each experiment the current given for the electromagnet is varied and for some values of electric current, the balls gets shuffled.

Always the characters written on the ball are read from the postion where the experimenter is standing. Given the number of balls taken for experimentation, characters written on the balls in the intial configuration and the characters written on the balls after rotation for ‘m’ seconds, write a program to check if the balls are shuffled or not.

For example, if six balls were taken for experimentation and the characters written on the balls intially are ‘p’, ‘a’, ‘s’, ‘g’, ‘h’, ‘y’ and the characters on the balls after rotation is ‘h’, ‘y’, ‘p’, ‘a’, ‘s’ ‘g’ then the balls haven’t shuffled. And if the characters written on the balls initially are ‘a’, ‘c’, ‘e’, ‘i’, ‘m’, ‘n’ and the characters on the balls after rotation are ‘i’, ‘n’, ‘m’, ‘a’, ‘c’, ‘e’ then the balls are shuffled.

**Input Format**

First line contains the number of balls taaken for experimentation ‘n’

Next ‘n’ lines contain the characters written on the balls initially

Next ‘n’ lines contain the characters written on the balls after rotating for ‘m’ seconds

**Output Format**

Print either Shuffled or Not Shuffled

**ANS.**

**from collections import \***

**n=int(input()); r\_s=[]**

**p=deque([input() for i in range(n)])**

**p1="".join([input() for i in range(n)])**

**for i in range(1,len(p)+1):**

**p.rotate(-i)**

**r\_s.append("".join(p))**

**if p1 in r\_s: print("Not Shuffled")**

**else: print("Shuffled")**

**Number of Cells taken for Cell Division**

A group of scientists are conducting an experimentation for six days on a particular type of cell ‘t’. These cells get the capability of cell division only when exposed to fresh air. The cells inside the test bed which have attained the capabilty of cell division divides into two every day morning by 5 am.

On a fine afternoon by 12:30 pm, the senior scientist of the group gives a test bed with one cell of type ‘t’, then the junior scientists collected ‘m’ cells of type ‘t’ and put them in the test bed. On the second day evening, one of the junior scientist opens the test bed and removes three cells from it. The test bed was left undisturbed for rest of the days. It was observed that ‘p’ cells were there inside the test bed on the evening of the sixth day. Given the number of cells in the test bed at the end of experimentation, write a C/C++ code to find the number of cells ‘m’ collected by the junior scientists.

For example, if 35 cells were in the test bed at the end of experimentation then the number of cells collected by junior scientist is 4.

**Input Format**

First line contains the number of cells p, left in the test bed at the end of experimentation

**Output Format**

Print the number of cells collected by junior scientist for experimentation, m

**ANS.**

#### **Palindrome and Repdigit**

Given a range of integers, write a C/C++ program to identify numbers which are not palindromes and whose cube root is a repdigit. A repdigit is a number which is composed of same digit. For example, tthe numbers between 10 and 99999 (inclusive) that are satisfying the conditions are 10648, 35937 and 85184 and their corressponding cube roots which are repdigits are 22, 33 and 44.

**Boundary conditions**

Upper bound of range of values can be 1013

**Input Format**

First line contains the lower bound value of the range, l

Next line contains the upper bound value of the range, u

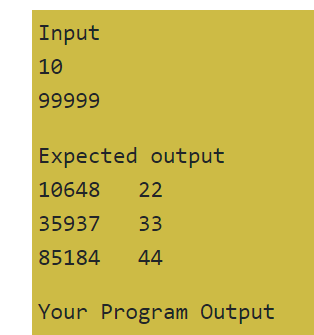
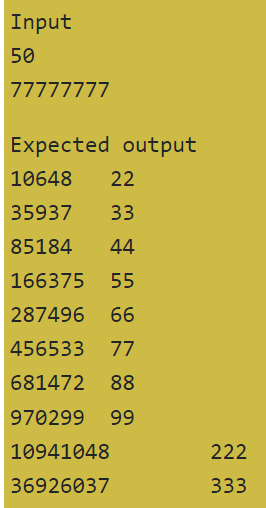
**Output Format**

In one line print the number ‘n’ which is not a palidrome and its cube root which is a repdigit separated by a tab

Print in ascending order of n

**Note:**

Cuberoot of a number can be found using cbrt function in math.h

Wrong code for ref:

#include <stdio.h>

#include <math.h>

int reverse(int n){

int rev=0,rem;

while(n!=0){

rem=n%10;

rev=rev\*10+rem;

n/=10;

}

return rev;

}

int main(){

int l,u;

scanf("%d\n%d",&l,&u);

for(int i=l;i<=u;i++){

if(i%11==0 && pow(i,3)<u+1 && reverse(pow(i,3))==pow(i,3)){

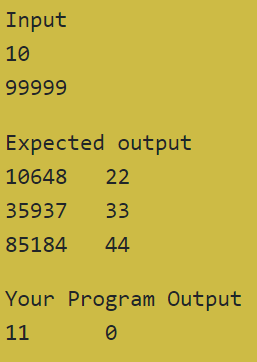
printf("%d\t%d",pow(i,3),i);

}

}

}

Output:



#### **Find Fruits Left Over**

Fruit smash is an interesting online game in which three fruits apple, orange and mango are arranged in a linear fashion. If all the three different fruits occur adjacent then they are smashed. The smashing process continue till no three adjacent fruits are different. The smashing process starts from the left side. A single smash at point 'p' may initiate many subsequent smashes to the points 'q' < 'p' and all triplets on the left side must be smashed before the triplets on their right side. In the representation, letter ‘a’ is used for apple, ‘m’ for mango and ‘o’ for orange.

Given a string of letters representing the arrangement of fruits, write a C/C++ program to findout the fruits that are left over after the smashing process. If all the fruits are smashed then print ‘No fruit is left over’.

For example, if the input is amammoamaoo then the output is am representing that an apple and a mango is left over. If the input is momoaamoomaa then output should be ‘No fruit is left over’.

**Boundary Conditions**

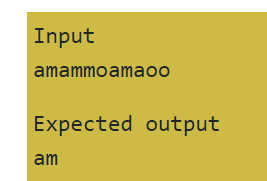
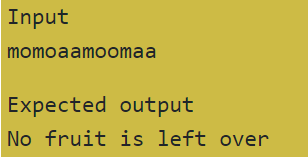
The length of the string can be upto 200

**Input Format**

First line contain the string representing the arrangement of fruits. One letter for one fruit, ‘a’ for apple, ‘o’ for orange and ‘m’ for mango

**Output Format**

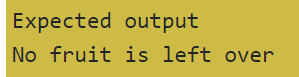
Print the letters of the fruits left over and if all the fruits are smashed then print ‘No fruit is left over’

** **

INPUT:

mamoaamamomaamooooaoammmmoaamamomaamooooaoammmaooamoaamamomaamooooaoammmmoaamamomaamooooaoammmaooamoaamamomaamooooaoammmmoaamamomaamooooaoammmaooamoaamamomaamooooaoammmmoaamamomaamooooaoammmaooao

OUTPUT:



#### **Identify Powers of Three**

Given an integer n, write a C++ program to find m where m is maximum of (n, n-1, n-2) and m can be expressed as sum of power(3, k) where k>=1. For example, if n is 123 then m is 123 and it can be written as 81, 27, 9, 3, 3. If n is 257, then m is 255 and it can be written as 243, 9, 3.

**Input Format**

First line contains the number, n

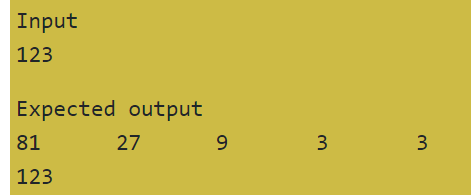
**Output Format**

First line contains powers of 3 which would be used for writing m separated by a tab (Print Powers of 3 from higher to lower)

Next line contains the value of m that can be expressed as sum of power of 3

**Note:**

In the expected output, there is a tab character is present at the end of the first line



#### **Plants Grown and Amount of Water**

**Note: There were some mistakes in the description of the question and now it is rectified. Apologies for that**

A farmer does drip irrigation in his field and every plant in his field would consume 1ml of water every alternate day. On a fine day, he plants ‘p’ plants and sows ‘s’ seeds in the field. The seeds are so special that they grow into plants two days. The farmer sows ‘s’ seed every alternate day. Given the number of plants he planted on the first day, the number of seeds he sows every alternate day and the value of ‘n’, write a C/ C++ program to find the number of plants that would be in his field on nth day and the total amount of water consumed by his plants. For example, if he plants four plants, sows five seeds every alternate day then on the third day, there will be totally 9 plants in his field and the total water consumed will be 13 ml and on the seventh day there will be 19 plants and total water consumed will be 46 ml.

**Input Format**

First line contains the number of plants he planted, p

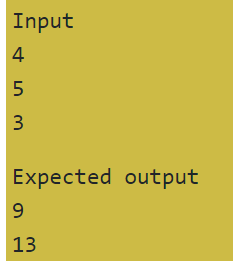
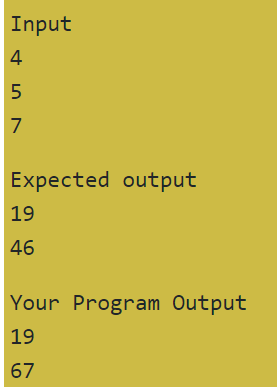
Next line contains the number of seeds he sow every alternate day, s

Next line contains the value of ‘n’

**Output Format**

First line should contain the total number of plants in his field on nth day

Next line should contain the total quantity of water consumed by the plants in his field

****

**p,s,n=int(input()),int(input()),int(input())**

**w=p**

**for i in range(3,n+1):**

**if i%2!=0: p+=s; w+=p**

**print(p,w,sep="\n")**

**C CODE:**

**#include <stdio.h>**

**#define ll long long**

**void solve(ll \*p,ll \*s,ll \*n){**

**ll w=\*p;**

**for(ll i=3;i<\*n+1;i++){**

**if (i%2!=0){**

**\*p+=\*s;**

**w+=\*p;**

**}**

**}**

**printf("%ld\n%ld",\*p,w);**

**}**

**int main(){**

**ll p,s,n;**

**scanf("%ld\n%ld\n%ld",&p,&s,&n);**

**solve(&p,&s,&n);**

**return 0;**

**}**

#### **Miscommunication from Mars**

Space scientist is communicating a group of ‘n’ numbers from Mars where ‘n’ is an even number. The ‘n’ numbers are formed from a set of numbers ‘s’ and each number is paired with itself. But the numbers are not ordered. That is if six numbers are to be communicated and they are formed from two numbers 3 and 4 then they can be 3 4 3 4 3 3. But during communication one of the numbers ‘n1’ is wrongly communicated as ‘n2’, both n1 and n2 are from ‘s’.

Given the difference between sum of numbers at the sending side and sum of the set of numbers at the receiving side and XOR of values at the receiving end, write a C++ program to identify the numbers that might have miscommunicated. For example, given that s = {2, 11, 4, 3, 7, 9} and difference in sum as 1 and XOR of the data at the receiving end as 1 then the numbers that might be miscommunicated are 2 or 3. And if s = {2, 11, 4, 3, 7, 9} and difference in sum as 2 and XOR of the data at the receiving end as 14 then the numbers that might be miscommunicated are 7 or 9.

The input is given such that only two numbers have chance of getting miscommunicated.

**Input Format**

First line contains the number of elements in set s, n

Next ‘n’ lines contain the numbers in set s

Next line contains the difference in sum, d

Next line contains the XOR value of data at the receiving end

**Output Format**

Print the two numbers that might have got miscommunicated in sorted order in a single line separated by a tab

**ANS.**

**s=sorted([int(input()) for i in range(int(input()))])**

**d,x=int(input()),int(input())**

**for i in range(len(s)):**

**for j in range(i+1,len(s)):**

**if s[i]^s[j]==x and abs(s[i]-s[j])==d: print(s[i],s[j],sep="\t")**

**C CODE:**

**#include <stdio.h>**

**#include <math.h>**

**//SELECTION SORT**

**void sort(int arr[],int n){**

**for (int i=0;i<n;i++){**

**for (int j=0;j<n-i-1;j++){**

**if (arr[j]>arr[j+1]){**

**int temp=arr[j];**

**arr[j]=arr[j+1];**

**arr[j+1]=temp;}}}**

**}**

**int main(){**

**int n,d,x;**

**scanf("%d",&n);**

**int s[n];**

**for(int i=0;i<n;i++){scanf("%d\n",&s[i]);}**

**sort(s,n);**

**scanf("%d",&d);**

**scanf("%d",&x);**

**for(int i=0;i<n;i++){**

**for(int j=i+1;j<n;j++){**

**//when using xor in C remember to put brackets around the operands before the '==' sign**

**if (abs(s[i]-s[j])==d && (s[i]^s[j])==x){**

**printf("%d\t%d\n",s[i],s[j]);}**

**}}**

**return 0; //NOT NECESSARY**

**}**

#### **Packing Water Bottles for a Resort**

There is a resort ‘ABC’ in a hill station, during season time many tourist come and stay in the resort. Everyday each resident of the resort is given a water bottle. From a mineral water plant, water bottles are packed in a cardboard box and sent to the resort. As per the bookings done, during first ‘m’ days the number of tourists staying in the resort gets increased and during the next ‘n’ days the number of tourists staying in the resort gets decreased. The resort places the order to the mineral water plant well in advance. There are two packaging machines M1 and M2 in the mineral water plant. M1 is used for packing when the number of bottles are multiple of 5 and M2 is used for packing all number of bottles. The working speed of M1 is much faster than M2 and both the machines pack the bottles faster when they are given number of bottles in ascending order. Given the number of guests for the first ‘m’ and subsequent ‘n’ days, write a C/C++ program to find the ordered list of number of bottles to be sent to M1 and M2 so that the packing can be done in minimum possible time.

For example, if the number of visitors for the first seven days is 12, 17, 20, 27, 29, 31, 35 and for the next six days the number of visitors in 30, 28, 25, 23, 16, 5 then the ordered list of bottles to be packed given for M1 is 5, 20, 25, 30, 35 and for M2 it is 12, 16, 17, 23, 27, 28, 29, 31.

**Input Format**

First line contains the value of m

Next line contain ‘m’ values separated by a space

Next line contains the value of n

Next line contain the value of ‘n’ values separated by a space

**Output Format**

First line should contain the number of bottles to be packaged by M1 in order

Next line should contain the number of bottles to be packaged by M2 in order

**Note:**

There is a space at the end of both the lines in the expected output

**ANS.**

**m=int(input())**

**bm=list(map(int,input().split()))**

**n=int(input())**

**bn=list(map(int,input().split()))**

**x=sorted(bm+bn); m1=[]; m2=[]**

**for i in x:**

**if i%5==0: m1.append(i)**

**else: m2.append(i)**

**print(\*m1,"")**

**print(\*m2,"")**

**C CODE:**

**#include <stdio.h>**

**void sort(int arr[],int n){**

**for(int i=0;i<n;i++){**

**for(int j=0;j<n-i-1;j++){**

**if (arr[j]>arr[j+1]){**

**int temp=arr[j+1];**

**arr[j+1]=arr[j];**

**arr[j]=temp;}}}**

**}**

**int main(){**

**int m,n;**

**scanf("%d",&m);**

**int b[m];**

**for(int i=0;i<m;i++){scanf("%d",&b[i]);}**

**scanf("%d",&n);**

**int a[n];**

**for(int i=0;i<n;i++){scanf("%d",&a[i]);}**

**int c[m+n];**

**for(int i=0;i<m;i++){c[i]=b[i];}**

**for(int i=m,j=0;i<m+n;i++){c[i]=a[j++];}**

**sort(c,m+n);**

**for(int i=0;i<m+n;i++){if(c[i]%5==0){printf("%d ",c[i]);}}**

**printf("\n");**

**for(int i=0;i<m+n;i++){if(c[i]%5!=0){printf("%d ",c[i]);}}**

**}**

#### **Find the Five Numbers**

Consider five numbers A, B, C, D and E where A ≤ B ≤ C ≤ D ≤ E, given the sum of all the possible two numbers combination in ascending order, write a C program to find the numbers A, B, C, D and E. That is if the sum of all the possible two numbers combination is 110, 112, 113, 114, 115, 116, 117, 118, 120, and 121. Then the numbers are 54, 56, 58, 59, 62.

**Input Format**

First line contains ten numbers separated by a space representing the sum of possible two number combinations in ascending order

**Output Format**

Print the values of A, B, C, D and E separated by a space

LOGIC EXPLAINED- https://wild.maths.org/pairwise-puzzler

**ANS.**

**s=list(map(int,input().split()))**

**t=sum(s)//4**

**a=s[0]+s[1]+s[9]-t**

**b=t-s[1]-s[9]**

**c=t-s[0]-s[9]**

**d=t-s[0]-s[8]**

**e=s[0]+s[8]+s[9]-t**

**print(a,b,c,d,e)**

#### **Lucky card**

An ordered letter pattern is a set of letters in alphabetical order. In a lucky game, a player gets a letter card which has got a matrix of alphabets in it. The letter card has N rows of M random letters in each row. An ordered letter pattern P of length L <= Min(N,M) will be announced.

If the letters of ‘P’ occurs continuously in the ordered letter pattern that could be formed with letters either along row or column in the letter card of the player then the player is lucky.

Given the matrix in the letter card of the player and and an ordered letter pattern P, develop pseudocode and C code to print “Lucky Player”. Otherwise, print “Unlucky Player”

For example, if the player has a letter card that has the following matrix

Q W E R

A S D F

Z X C V

and the announced ordered pattern P is AFS, then the player is lucky because the ordered letter pattern that could be formed with ASDF matches with the given pattern P.

**Input Format**

Enter the number of letter collection, N

Enter the number of letters M in each collection

Enter N\*M characters one in each row

Enter the pattern P

**Output Format**

Ordered letters in collection that matches with ordered letters in pattern

Lucky Player

or

Unlucky Player

**ANS.**

**from itertools import \***

**#function for transpose of matrix in python**

**def transpose(a,n,m):**

**x=[]**

**for i in range(m):**

**b=[]**

**for j in range(n):**

**b.append(a[j][i])**

**x.append(b)**

**return x**

**n,m=int(input()),int(input())**

**mat=[input().split() for i in range(n)]**

**p=input()**

**mat\_t=transpose(mat,n,m)**

**w=["".join(sorted(i)) for i in mat]**

**for i in mat\_t: w.append("".join(sorted(i)))**

**flag=0**

**for i in w:**

**comb=list(combinations(i,len(p)))**

**for j in comb:**

**if "".join(j)==p:**

**print(i)**

**print("Lucky Player")**

**flag=1**

**break**

**if flag==0: print("Unlucky Player")**

**WITH EXIT FUNCTION:**

**from itertools import \***

**#code for transpose of matrix in python**

**def transpose(a,n,m):**

**x=[]**

**for i in range(m):**

**b=[]**

**for j in range(n):**

**b.append(a[j][i])**

**x.append(b)**

**return x**

**n,m=int(input()),int(input())**

**mat=[input().split() for i in range(n)]**

**p=input()**

**mat\_t=transpose(mat,n,m)**

**w=["".join(sorted(i)) for i in mat]**

**for i in mat\_t: w.append("".join(sorted(i)))**

**for i in w:**

**comb=list(combinations(i,len(p)))**

**for j in comb:**

**if "".join(j)==p:**

**print(i)**

**print("Lucky Player")**

**exit()**

**print("Unlucky Player")**

#### **Last M Digit Generate-able**

A number ‘f’ is said to be generatable if it can be generated in a Fibonacci-like sequence from its digits. Initially the sequence contains the digits in it from left to right and the next number in the sequence is generated by summing up the last ‘d’ numbers in the sequence where ‘d’ is the number of digits in ‘f’. For example, the number 197 can be got in a sequence 1, 9, 7, 17, 33, 57, 107, 197 by adding last three numbers.

A number ‘n’ is said to be last ‘m’ digit generate-able if the number that can be formed with last ‘m’ digits of it can be generated from them. For example, 23197 is last 3 digit generate-able, 371104 is last 4 digit generate-able, 17844121607 is last 8 digit generate-able.

**Input Format**

First line contains the number n

Next line contains the number of last digits m

**Output Format**

Print either Generated or Cannot be generated

**ANS.**

**n,m=int(input()),int(input())**

**y=str(n)[len(str(n))-m:]**

**x=[int(i) for i in y]**

**while(x[-1]<int(y)):**

**x.append(sum(x))**

**x.pop(0)**

**if x[-1]==int(y): print("Generated")**

**else: print("Cannot be generated")**

**C++ CODE:**

**//LEARN ABOUT DEQUE AND ACCUMULATE FUNCTION**

**//(WORKS LIKE REDUCE IN PYTHON)**

**//ALSO SEE HOW TO CHANGE INT KEYWORD TO MEAN LONG LONG INT**

**#include <iostream>**

**#include <deque>**

**#include <cmath> //math.h => cmath in C++**

**#include <numeric>**

**#define int long long**

**using namespace std;**

**int sum\_func(int x,int y){return x+y;}**

**int32\_t main(){**

**deque <int> num;**

**int n,m,j=0,num1=0,sum=0;**

**cin>>n;**

**cin>>m;**

**while(m!=0){**

**int d=n%10;**

**num1+=d\*(pow(10,j++));**

**num.push\_front(d);**

**n/=10;**

**m--;**

**}**

**while(num[num.size()-1]<num1){**

**num.push\_back(accumulate(num.begin(),num.end(),sum,sum\_func));**

**num.pop\_front();**

**}**

**if(num[num.size()-1]==num1){cout<<"Generated";}**

**else{cout<<"Cannot be generated";}**

**}**

#### **Color of Building Blocks**

There is a competition held among students of class six to build a given structure using building blocks. Blocks of ‘n’ different colors are given to the participants. One of the rule of the competition states that a block of a particular color can be chosen only after choosing ‘m’ other given colors where m≤n-1. Given the ‘n’ colors, the value of ‘m’ and the choice of colors made by the participants, write a C++ program to check if the choices made

For example given four color pink, yellow, blue and green represented by letters ‘p’, ‘y’, ‘b’ and ‘g’ and the value of m as two, it is valid to choose the colors in the sequence pbygpbygbpgypgbygp and the choice of color pbygygbpbygpbygbpgypgbygp is invalid for m=2.

**Boundary Conditions**

Maximum number of building blocks used in the competition is 300

n ≤ 20

Colors are represented by a unique letter

**Input Format**

First line contains the value of n

Next line contain the n colors used by the participant and separated by a space

Next line contains the value of m

Next line contains the colors used by the participants in order

**Output Format**

Print Valid or Invalid

**ANS.**

**n=int(input())**

**colors=input().split()**

**m=int(input())**

**seq=input()**

**for i in range(len(seq)):**

**if seq[i] in seq[i+1:i+m+1]: print("Invalid"); exit()**

**print("Valid")**

**C CODE:**

**#include <stdio.h>**

**#include <stdlib.h> // for exit function**

**#include <string.h>**

**int main(){**

**int m,n;**

**scanf("%d",&n);**

**char colors[n],seq[1000];**

**// how to input with spaces in between elements**

**for(int i=0;i<2\*n;i++){ //2\*n for blank spaces**

**char x;**

**scanf("%c",&x);**

**if (x!=32){colors[i]=x;} //32 IS THE ASCII FOR BLANK SPACE**

**}**

**scanf("%d",&m);**

**scanf("%s",&seq);**

**// alternate code for string slicing approach (can be applied at many places)**

**for (int i=0;i<strlen(seq);i++){**

**for (int j=i+1;j<i+m+1;j++){**

**if(seq[i]==seq[j]){**

**printf("Invalid");**

**exit(0); //exit function in C**

**}**

**}**

**}**

**printf("Valid");**

**}**

#### **Minimum Value Expression**

Given three integers a,b,c ≥ 0, write a C++ program to find the minimum value that may be obtained by expression with only binary operators +, - and \*. Use all the three integers for finding minimum value. For example when a= 2, b =1 c = 3 minimum value is -5

**Input Format**

First line contains the value of a

Next line contains the value of b

Next line contains the value of c

**Output Format**

Print the minimum value that can be obtained

**ANS.**

**a,b,c=int(input()),int(input()),int(input())**

**nums=sorted((a,b,c))**

**if nums[1]\*nums[2] >= nums[1]+nums[2]: print(nums[0]-(nums[1]\*nums[2]))**

**else: print(nums[0]-(nums[1]+nums[2]))**

**C CODE:**

**#include <stdio.h>**

**#define int long long**

**\_\_int32\_t main(){**

**int a,b,c;**

**scanf("%lld\n%lld\n%lld",&a,&b,&c);**

**int arr[]={a,b,c};**

**for(int i=0;i<3;i++){**

**for(int j=0;j<2-i;j++){**

**if(arr[j]>arr[j+1]){**

**int temp=arr[j];**

**arr[j]=arr[j+1];**

**arr[j+1]=temp;}}}**

**if((arr[1]+arr[2])>=(arr[1]\*arr[2])){printf("%lld",arr[0]-(arr[1]+arr[2]));}**

**else{printf("%lld",arr[0]-(arr[1]\*arr[2]));}**

**}**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**vector <int> nums;**

**for (int i=0;i<3;i++){int x; cin>>x; nums.push\_back(x);}**

**sort(nums.begin(),nums.end());**

**if ((nums[1]+nums[2])>=(nums[1]\*nums[2]))**

**{cout<<nums[0]-(nums[1]+nums[2]);}**

**else {cout<<nums[0]-(nums[1]\*nums[2]);}**

**}**

**Or alternatively:**

**//we can define the length of vector**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**vector <int> nums(3);**

**for (int i=0;i<nums.size();i++){cin>>nums[i];}**

**sort(nums.begin(),nums.end());**

**if ((nums[1]+nums[2])>=(nums[1]\*nums[2]))**

**{cout<<nums[0]-(nums[1]+nums[2]);}**

**else {cout<<nums[0]-(nums[1]\*nums[2]);}**

**}**

#### **Correctness of Printed Sheets**

Number labels are to be prepared by a small scale printing company. The sheets are cut into uniform dimension and fed into the printing machine. The printing machine has to print some numbers in ascending order.

****

The printing machine doesn’t do the job always correct it print numbers in wrong order sometimes. But some of the printed sheets are easily correctable by a single cut and a single paste. For example, if the printed sheet has numbers 17, 19, 23, 34, 56, 71, 80 then it has done the job correct. If the printed sheet has the numbers 76, 89, 91, 96, 23, 28, 29, 37, 45 then it is easily correctable since by a single cut after number 96 and single paste before 76 will contain the numbers in order that is cut after the 4th number in the printed sheet will do. If the printed sheets has the numbers 76, 89, 91, 96, 23, 28, 97, 29, 37, 45 is not easily correctable. Given the numbers in a printed sheet, write a C++ code to categorize it to Correct, Easily correctable or Not easilt correctable.

**Input Format**

First line contains the count of numbers printed in the sheet

Next line contain the numbers printed on the sheet separated by a space

**Output Format**

Print either Correct, Easily correctable or Not easily correctable

If the sheet is easily correctable then print the position of the number after which cut has to be made in the next line

**ANS.**

**from collections import \***

**n=int(input())**

**nums=list(map(int,input().split()))**

**nums1=sorted(nums)**

**flag=0**

**for i in range(n):**

**if nums[i]!=nums1[i]: flag=1; break**

**if flag==0: print("Correct")**

**if flag==1:**

**possible\_combs=[]; diff=[]**

**x=deque(nums1)**

**for i in range(n-1):**

**diff.append(i+1)**

**x.rotate(-1)**

**possible\_combs.append(list(x))**

**if nums in possible\_combs:**

**print("Easily correctable")**

**print(n-diff[possible\_combs.index(nums)])**

**else: print("Not easily correctable")**

**C++ CODE:**

**#include <iostream>**

**#include <algorithm>**

**#include <deque>**

**#include <cstdlib>**

**using namespace std;**

**#define int long long**

**int check(deque <int> l,deque <int> m){**

**int flag=1;**

**for(int i=0;i<l.size();i++){**

**if (l[i]!=m[i]){flag=0; return flag;}**

**}**

**return flag;**

**}**

**deque <int> rotate(deque <int> l){**

**int x=l[l.size()-1];**

**l.pop\_back();**

**l.push\_front(x);**

**return l;**

**}**

**deque <int> copy(deque <int> l){**

**deque <int> x;**

**for(int i=0;i<l.size();i++){**

**x.push\_back(l[i]);**

**}**

**return x;**

**}**

**int32\_t main(){**

**int n;**

**cin>>n;**

**deque <int> nums(n);**

**for(int i=0;i<n;i++){cin>>nums[i];}**

**deque <int> nums1=copy(nums);**

**sort(nums.begin(),nums.end());**

**if(check(nums1,nums)==1){**

**cout<<"Correct";**

**exit(0);**

**}**

**int flag=0;**

**for(int i=0;i<n-1;i++){**

**nums=rotate(nums);**

**if(check(nums1,nums)==1){**

**flag=1;**

**cout<<"Easily correctable"<<"\n"<<i+1;**

**break;**

**}**

**}**

**if(flag==0){cout<<"Not easily correctable";}**

**}**

#### **Neutralizing Solution**

A chemist tries to neutralzie a collection ‘C’ of solutions of different strength in ‘n’ cups. The task of the chemist is to find a neutralizing solution ‘X’ of strength ‘S’. In this task, the chemist will be given solutions in cups that are having strength that are integers and the strength of the neutralizing solution can be also only an integer.

In the process of neutralization he will be given a bigger vessel ‘B’ and he has to repeat the following steps:

1. For each cup Ci in C, pour solution ‘X’ into Ci. Strength of the combined solution in Ci will be S – Si,where Siis the strength of the original solution in Ci

2. For each cup Ci in C, pour the solution in Ci into ‘B’

3. Strength of the solution in ‘B’ is sum of the strength of the solutions in Ci

After this process if the strength of the solution in ‘B’ is zero then the solution ‘X’ is the neutralizing solution. Not all collection of solutions have a neutralizing solution. For example, if there are three cups with strength of solution as 1, 2 and 3 then the strength of the required neutralizing solution is 2.

Given the strength of the solutions in the ‘n’ cups, write a C++ program to check if there exist an solution ‘x’ that can neutralize them and determine the strength of the neutralizing solution if it exist.

**Input Format**

First line contains the value of n

Next line contains the strength of solutions in the cup and separated by a space

**Output Format**

In the frist line, print Yes if a neutralizing solution exist and No otherwise

In the next line, print the strength of the neutralizing solution if exists

**ANS.**

**n=int(input())**

**cups=list(map(int,input().split()))**

**for i in range(min(cups),max(cups)+1):**

**x=sum([j-i for j in cups])**

**if x==0: print("Yes"); print(i); exit()**

**print("No")**

**C CODE:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define int long long**

**//HOW TO RETURN TWO VALUES FROM A FUNCTION**

**//PASS TWO BLANK VARIABLES AS PARAMETERS AND FILL THEM IN THE FUN.**

**void max\_min(int arr[],int n,int max,int min){**

**for(int i=0;i<n;i++){**

**for(int j=0;j<n-i-1;i++){**

**if(arr[j]>arr[j+1]){**

**int tmp=arr[j];**

**arr[j]=arr[j+1];**

**arr[j+1]=tmp;}}}**

**max=arr[n-1];**

**min=arr[0];**

**}**

**\_\_int32\_t main(){**

**int n;**

**scanf("%lld",&n);**

**int arr[n];**

**for(int i=0;i<n;i++){scanf("%lld",&arr[i]);}**

**int max,min; //BLANK VARIABLES**

**max\_min(arr,n,max,min); //PASSED IN FUNCTION**

**for(int i=min;i<max+1;i++){**

**int sum=0;**

**for(int j=0;j<n;j++){sum+=(arr[j]-i);}**

**if(sum==0){printf("Yes\n%lld",i); exit(0);}**

**}**

**printf("No");}**

#### **Number of Unique Points**

Given ‘n’ integers which can be value of X or Y coordinate, write a C++ program to find the number of unique points that can be formed. For example given three values 1, 1 and 2 there can be four unique points (1, 1), (1, 2), (2, 1) and (2, 2) and when the three values given are 1, 2 and 3 there can be nine unique points (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2) and (3, 3). Given the value of ‘n’ integers, write a C++ program to find the number of unique points that can be formed with the given points.

**Boundary Conditions**

n can both positive and negative integers

**Input Format**

First line contains the value of n

Next line contains the ‘n’ integers separated by a space

**Output Format**

Print the number of unique points that can be formed

**ANS.**

**n=int(input())**

**nums1=[]**

**nums=list(map(int,input().split()))**

**for i in nums:**

**if i not in nums1: nums1.append(i)**

**print(len(nums1)\*\*2)**

**print(\*nums1)**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**#include <cmath>**

**using namespace std;**

**int main(){**

**int n;**

**cin>>n;**

**vector <int> nums(n);**

**vector <int> nums1;**

**for(int i=0;i<n;i++){cin>>nums[i];}**

**for(int i=0;i<n;i++){**

**if (find(nums1.begin(),nums1.end(),nums[i])!=nums1.end()){continue;}**

**else {nums1.push\_back(nums[i]);}**

**}**

**cout<<pow(nums1.size(),2)<<"\n";**

**for(int i=0;i<nums1.size();i++){cout<<nums1[i]<<" ";}**

**}**

#### **Check Money Taken by Child**

A child wants to go to a chocolate expo and buy some chocolates. His father gives him a task and say that he will take the child to the expo provided he successfully does the task. The chocolate expo has variety of chocolates in all the prices which are natural numbers. The father gives a bowl of coins and tells the child that two rupees has to be spent as bus fare to reach the expo and he will buy four chocolates of same cost for him provided the child can pick coins that sum up to value with which they can reach the chocolate expo and buy exactly four chocolates of same cost and no money should be left out after purchasing the chocolates. For example, if the input is 110 (decimal value is 6) then print Success and if the string is 100 (decimal value is 4) then print Fail.

Given a string that gives binary representation of the money taken by the child, write a C++ code to check if the child has succeeded in the task or not.

**Boundary Conditions**

1 <= l <= 300, where l is the length of the binary string

Decimal value of the binary string > 0

**Input Format**

First line contains the binary representation of the amount taken by the child

**Output Format**

Print Success if the amount satisfies the constraints of the father and print Fail otherwise

**ANS.**

**print("Success" if (int(input(),2)-2)%4==0 else "Fail")**

**C++ CODE:**

**#include <iostream>**

**using namespace std;**

**#define int long long**

**int decimal\_convertor(string n){**

**int base=1,dec=0;**

**for(int i=n.length()-1;i>=0;i--){**

**if (n[i]=='1'){dec+=base;}**

**base\*=2;**

**}**

**return dec;**

**}**

**int32\_t main()**

**{**

**string binary;**

**cin>>binary;**

**int decimal=decimal\_convertor(binary);**

**if((decimal-2)%4==0) {cout<<"Success";}**

**else {cout<<"Fail";}**

**}**

**C CODE:**

**#include <stdio.h>**

**#include <string.h>**

**#define int long long**

**int decimal\_convertor(char n[]){**

**int base=1,dec=0;**

**for(int i=strlen(n)-1;i>=0;i--){**

**if(n[i]=='1'){dec+=base;}**

**base\*=2;**

**}**

**return dec;**

**}**

**\_\_int32\_t main(){**

**char binary[301];**

**scanf("%[^\n]%\*c",&binary);**

**int decimal=decimal\_convertor(binary);**

**if((decimal-2)%4==0){printf("Success");}**

**else{printf("Fail");}**

**}**

#### **Area Watered by Diesel Water Pump**

A farmer uses diesel water pumps for irrigating his fields. The pumps start only when it is filled with ‘x’ litres of diesel and it can water ‘y’ acres of field. When the diesel level becomes one third of ‘x’ it stops working. Everyday the farmer gets ‘z’ litres of diesel. For example, if the value of x is 3, y is 2 and z is 12 then 10 acres can be irrigated at the maximum. Given the value of ‘x’ and ‘y’, write a C++ program to find out the maximum area that can be watered with ‘z’ litres of diesel.

**Boundary Conditions**

x, y, z are integers and z is a multiple of x

**Input Format**

First line contains the number of days, n

Next ‘n’ lines contains the value of x, y, z separated by a space

**Output Format**

In one line print the total acres that can be irrigated on everyday

**ANS.**

**for i in range(int(input())):**

**x,y,z=map(int,input().split())**

**area=0**

**while z>=x:**

**area+=y**

**z-=((2/3)\*x)**

**if round(z)==x: area+=y**

**print(area)**

#### **Minimum Bin-Distance Numbers**

Two numbers n1 and n2 are said to be have a bin-distance of ‘x’ if ‘x’ bits are different in the corressponding positions in binary representation of n1 and n2. For example, numbers 4 (100) and 6 (110) are having a distance of 1 whereas numbers 2027 (11111101011) and 119 (1110111) are having a distance of 7.

Given a collection ‘c’ of ‘n’ numbers and a number ‘m’, write a C++ program to find the number(s) in ‘c’ that is having minimum bin-distance with m. For example, given three numbers 4, 2 and 5 in collection ‘c’ and if m = 6 then numbers 4 and 2 are having a minimum distance of 1 with 6.

**Hint:**Try to solve this problem without doing decimal to binary conversion

**Input Format**

First line contains the number of integers ‘n’ in C

Next ‘n’ lines contain the numbers in C

Next line contains the number m

**Output Format**

Print all the elements having minimum bin-distance with m. Print one number in one line. Print the minimum bin-distance elements in the order in which it was given in the input

**ANS.**

**def set\_bits(n):**

**count=0**

**while n!=0:**

**n=n&(n-1)**

**count+=1**

**return count**

**nums=[int(input()) for \_ in range(int(input()))]**

**m=int(input())**

**dis=[set\_bits(m^i) for i in nums]; min\_dis=min(dis)**

**for i in range(len(nums)):**

**if dis[i]==min\_dis: print(nums[i])**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**#define int long long**

**int set\_bits(int n){**

**int count=0;**

**while(n!=0){**

**n=n&(n-1);**

**count+=1;**

**}**

**return(count);**

**}**

**int32\_t main(){**

**int n,m;**

**cin>>n;**

**vector <int> nums(n);**

**for(int i=0;i<n;i++){cin>>nums[i];}**

**cin>>m;**

**vector <int> dis;**

**for(int i=0;i<n;i++){dis.push\_back(set\_bits(nums[i]^m));}**

**int min\_dis=\*min\_element(dis.begin(),dis.end()); //min function in stl**

**for(int i=0;i<n;i++){**

**if(dis[i]==min\_dis){cout<<nums[i]<<"\n";}**

**}**

**}**

#### **Arrange the Tiles**

In a tile game a nXn grid of tiles with numbers are given to the players. In each move then can pluck a tile in a particular square and fix it on some other square. Given the numbers written on the tiles in current configuration and the numbers written on the tiles in desired configuration, write a C++ program to determine the minimum number of moves to be made by the player to attain the desired configuration from the current configuration. For example, if the numbers on a 3X3 tile (read row wise) is 23, 12, 45, 67, 53, 11, 13, 90, 75 and the desired cofiguration is 11, 12, 13, 23, 45, 53, 67, 75, 90 then 6 moves are required.

**Input format**

First line contains the number of tiles, nXn

Next line contains the numbers written on the tiles in the current configuration separated by a space

Next line contains the numbers written on the tiles in the desired configuration separated by a space

**Output format**

Print the minimum number of moves to be made to reach desired configuration from initial configuration

**ANS.**

**n=int(input())**

**nums=list(map(int,input().split()))**

**nums\_desired=list(map(int,input().split()))**

**moves=0**

**for i in range(len(nums)):**

**if nums[i]!=nums\_desired[i]:**

**nums[nums.index(nums\_desired[i])],nums[i]=nums[i],nums[nums.index(nums\_desired[i])]**

**moves+=1**

**print(moves)**

**C++ CODE:**

**//storing the index of an element in vector and swap, find function in //algorithm library with two vectors declared in one line**

**#include <iostream>**

**#include <algorithm>**

**#include <vector>**

**using namespace std;**

**int main(){**

**int n,moves=0;**

**cin>>n;**

**vector <int> nums(n),nums\_desired(n);**

**for(int i=0;i<n;i++){cin>>nums[i];}**

**for(int i=0;i<n;i++){cin>>nums\_desired[i];}**

**for(int i=0;i<n;i++){**

**if(nums[i]!=nums\_desired[i]){**

**int x=find(nums.begin(),nums.end(),nums\_desired[i])-nums.begin();**

**swap(nums[x],nums[i]);**

**moves++;**

**}**

**}**

**cout<<moves;**

**}**

#### **Speed of a Boat**

**There was a mistake in description and there is a Change in Output Format**

**Apologies for that**

A water current, in a river is the flow of water influenced by gravity as the water moves downhill to reduce its potential energy. Upstream is a water current which acts against the direction of the boat movement and downstream is a water current which acts in the same direction of the boat movement. In upstream, a boat travels in the river from one side to the other side in ‘x’ hours and in downstream it takes ‘y’ hours.

For example, if the time taken in upstream is 3 and time taken in downstream is 4 then the time taken by the boat without water current will be

3.428

Write a C++ program to find the time taken by the boat to move from one side to another without water current

**Input Format**

First line contains the time taken by the boat in upstream

Next line contains the time taken by the boat in downstream

**Output Format**

Print time taken by boat

**Note**

Print floor of time taken by boat

**ANS.**

**print(int(2/(int(input())\*\*(-1)+int(input())\*\*(-1))))**

**C++ CODE:**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int main(){**

**int x,y;**

**cin>>x>>y;**

**cout<<floor(2/((pow(x,-1)+pow(y,-1))));**

**}**

#### **Stern-Brocot Sequence**

# **Stern-Brocot Sequence is a sequence similar to fibanocci series. A sequence ‘a’ is built using the following rules:**

* a0 = 0
* a1 = 1
* a2i = ai
* a2i+1 = ai + ai+1
* For example, the sequence of numbers 1, 1, 2, 1, 3 is a Stern-Brocot Sequence and 1, 1, 2, 1, 3, 2, 3, 3, 4 is not a Stern-Brocot Sequence since the eigth element is not as per rule. Given ‘n’ numbers write a C++ code to find out if it can be a Stern-Brocot Sequence when taken in order.

**Input Format**

First line contains the number of elements, n

Next line contains the elements separated by a space

**Output Format**

Print Yes if the elements form Stern-Brocot Sequence and print No and the index of the first element which violates the rule

**ANS.**

**GFG MODIFIED and MORE EFFICIENT:**

**def solve(n):**

**a=[1,1]**

**for i in range(1,n//2+1):**

**considered\_element=a[i]**

**precedent=a[i-1]**

**a.append(considered\_element+precedent)**

**a.append(considered\_element)**

**if n%2==0: a.pop(-1); a.pop(-1)**

**else: a.pop(-1)**

**return a**

**def check(l1,l2):**

**for i in range(len(l1)):**

**if l1[i]!=l2[i]: print("No",i+1,sep="\n"); exit()**

**print("Yes")**

**n=int(input())**

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

**check(a,solve(n))**

**ALTERNATE WAY (MY ORIGINAL WAY):**

**n=int(input())**

**a=dict(zip([i for i in range(1,n+1)],list(map(int,input().split()))))**

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

**if a[i]==0: print("No",i,sep="\n"); exit()**

**if (2\*i+1)<=n:**

**if (a[i+1]+a[i])!=a[2\*i+1]: print("No",2\*i+1,sep="\n"); exit()**

**if i%2==0:**

**if a[i//2]!=a[i]: print("No",i,sep="\n"); exit()**

**print("Yes")**

#### **Number of Connections in Mesh Topology**

In a mesh topology, every computer is connected to every other computers in the network. Cost of connecting adajcent computers (c1) is less when compared to connecting non-adjacent computers (c2). Given the number of nodes in the network cost of connecting adjacent nodes and cost of connecting non-adjacent nodes, write a C++ code to compute the total cost involved in forming the mesh topology.

For example, if the number of nodes in the network is 5 and c1 is 7 and c2 is 8 then the total cost is 75.

**Input Format**

First line contains the number of nodes in the network, n

Next line contains the cost of connecting adjacent computer, c1

Next line contains the cost of connecting non-adjacent computer, c2

**Output Format**

Print the total cost incurred in building the mesh topology for the network

**BASICALLY: NO. OF DIAGONALS\*C2+NO. OF SIDES\*C1**

**ANS.**

**n,c1,c2=int(input()),int(input()),int(input())**

**print(((n\*(n-3))//2)\*c2+n\*c1)**

**C++ CODE:**

**#include <iostream>**

**#include <cmath>**

**#define ll long long**

**using namespace std;**

**int main(){**

**int n,c1,c2;**

**cin>>n>>c1>>c2;**

**cout<<(ll)(floor(n\*(n-3)/2)\*c2+n\*c1);**

**}**

#### **Valid Customized Roman Numeral**

**Some modification is made in description and One More Example is added in the problem for Clarity**

A teacher wants to give an assignment on Roman numberals. But she is afraid that the students may get the answer from internet. Therefore she customizes the roman numerals with changes in the rule of formation. Romans represented numbers using the numerals I, V, X, L, C, D, and M as shown in the table:

|  |  |
| --- | --- |
| **Roman Numeral** | **Hindu-Arabic Equivalent** |
| I | 1 |
| V | 5 |
| X | 10 |
| L | 50 |
| C | 100 |
| D | 500 |
| M | 1000 |

There are a few rules for writing numbers with Roman numerals:

* Repeating a numeral represents addition of the number. For example, III represents 1 + 1 + 1 = 3.
* Only I, X, C, and M can be repeated that is III is valid but LL is not valid
* Writing numerals that decrease from left to right represents addition of the numbers. For example, LX represents 50 + 10 = 60 and XVI represents 10 + 5 + 1 = 16.
* Writing numerals that increase from left to right represents subtration. For example, IV represents 5 - 1 = 4 and IX represents 10 - 1 = 9.
* Maximum of two same small numerals can only be written to the left of a Roman numeral. For example, 8 can be represented by either VIII or IIX

For example, XXXIX is 39 and XXXXIIX is 48.  IIIIX is invalid. Given a string of roman numeral, write a C++ code to check if it is valid and also print the arabic equivalent if valid and print Invalid otherwise.

**More Example**

Value of LDMM = 1450

**Input Format**

First line contains the roman numeral string, s

**Output Format** : Print the arabic equivalent of s

**ANS.**

**STACK IMPLEMENTATION:**

**roman={'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000}**

**m=['I','V','X','L','C','D','M']**

**s=input()[::-1]**

**stack=[]**

**last\_index=0**

**for i in s:**

**if(m.index(i)<last\_index):**

**stack[-1]=stack[-1]-roman[i]**

**else:**

**last\_index=m.index(i)**

**stack.append(roman[i])**

**if 'VV' in s or 'LL' in s or 'DD' in s or len(set(s[1:min(4,len(s))]))==1:**

**print("Invalid"); exit()**

**print(sum(stack))**

**UNDERSTANDABLE CODE: REFER CODEAGE PDF AND C++ SOLN. VIT POD PDF**

#### **Find Number Up Side Down**

A number plate was painted in a workshop and fixed in a house. But the person who fixed the plate made a mistake and fixed the board up side down. Luckily the numbers written on the plate are readable up side down also. It is a two digit number. For example if difference is 72 then the numbers are 08 and 80.

Given the difference between the original number and the number read upside down, write a C++ code to find the numbers

**Boundary Conditions**

diff>0 and only one number has diff

**Input Format**

Difference between the original number and number read upside down

**Output Format**

Print the original number and number read upside down in ascending order separated by a space

**ANS.**

**diff=int(input())**

**digs=['0','1','6','8','9']**

**revs=['0','1','9','8','6']**

**nums=set()**

**for i in range(len(digs)):**

**for j in range(len(digs)):**

**nums.add(tuple(sorted([ digs[i]+digs[j] , revs[j]+revs[i] ])))**

**for i in nums:**

**if int(i[1])-int(i[0])==diff: print(i[0],i[1])**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**#include <iterator>**

**#include <set>**

**using namespace std;**

**int main(){**

**int diff;**

**cin>>diff;**

**vector <string> digs={"0","1","6","8","9"};**

**vector <string> revs={"0","1","9","8","6"};**

**set <pair<string,string>> nums;**

**for(int i=0;i<digs.size();i++){**

**for(int j=0;j<digs.size();j++){**

**pair <string,string> x;**

**x.first=digs[i]+digs[j];**

**x.second=revs[j]+revs[i];**

**nums.insert(x);**

**}}**

**//set traversal in c++ and stoi function to convert string numbers to int**

**//how to access the elements of a pointer to a pair**

**set <pair<string,string>> :: iterator it;**

**for(it=nums.begin(); it!=nums.end(); it++){**

**int d = stoi(it->second) - stoi(it->first);**

**if (d==diff) {cout<<it->first<<" "<<it->second;}**

**}**

**}**

#### **Cost of Watermelon**

A vendor had ‘x’ kg of watermelon in the morning and it had p1% of water. He couldn’t sell any watermelon till evening. In the evening, the percentage of water in the watermelon got decreased to p2%. Cost of one kg of watermelon was ‘c’. Find the total money the vendor would get if he can sell all the watermelon in the evening. For example, if there was 300 kg of watermelon in the morning and it had 97% of water in the morning and it had 90% of water in the evening and cost of 1kg of watermelon is Rs. 8 then the total money collected by the vendor will be Rs. 720.

**Input Format**

First line contains the weight of watermelon with the vendor in the morning, x

Next line contains the percentage of water in watermelon in the morning, p1

Next line contains the percentage of water in watermelon in the evening, p2

Next line contains the cost of one kg of watermelon, c

**Output Format**

Print the total money that would be got after selling all watermelon in the evening

**Important logical problem**

**ANS.**

**x,p1,p2,c=int(input()),int(input()),int(input()),int(input())**

**raw\_fruit=x\*(100-p1)/100**

**new\_weight=raw\_fruit\*100/(100-p2)**

**print(round(new\_weight\*c))**

**C++ CODE:**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int main(){**

**int x,p1,p2,c;**

**cin>>x>>p1>>p2>>c;**

**int raw\_fruit = x\*(100-p1)/100;**

**int new\_weight = raw\_fruit\*100/(100-p2);**

**cout<<floor(new\_weight\*c);**

**}**

#### **Common Points of Lines**

**Use Hint in the problem to solve it**

A set of lines have same ‘y’ coordinates so that is not significant for this problem. Points on those lines are represented by a triplet (a,b,k) where a is the x-coordinate of the starting point of the line, and b is the x-coordinate of the ending point of the line. The points on the line are a+k, a+2k, a+3k ... a+n\*k such that a<=a+k and a+nk<=b. For example, the points on the line represented by (1, 2, 0.3) are 1, 1.3, 1.6, 1.9. Given n lines (represented by the triplets), write a C++ code to print all points that are common to all the lines

**Boundary Conditions**

n<=10

a,b>=0

Number of points generated for each line <=50

**Input Format**

First line contains the number of lines, n

Next ‘n’ lines contain the start point, end point and the value of k separated by a space

**Output Format**

Print the points common to all lines in ascending order separated by a space

**Hint**

**Comparison of floating point values in C++ is very precise and sometimes gives unexpected results. To avoid that**

**When comparing two floating point values a and b**

**(i) Instead of writing a<b, write as a<b+0.001, here a is read from user and b is computed by an expression**

**(ii) Instead of writing a==b, write as fabs(a-b)<a\*0.001**

**ANS.**

**from math import fabs**

**from functools import reduce**

**pts=[]**

**for i in range(int(input())):**

**a,b,k=map(float,input().split()); l=set()**

**while(a<b+0.001):**

**l.add(round(a,1))**

**a+=k**

**pts.append(l)**

**x=sorted(list(reduce(lambda a1,b1:set.intersection(a1,b1),pts)))**

**for i in x:**

**if fabs(i-int(i))<a\*0.001: print(int(i),end=" ")**

**else: print(i,end=" ")**

#### **Varied Distance Relay Race**

A relay race is to be conducted by a sports club for ‘x’ meters. It is a different type of race in which each runner in a team can run different distances but there should be four runners in a team. There is a group of runners and they can run with maximum speed for varied distances according to their capacity. Given the value of ‘x’ and the speed with which each runner in the group can run check if four players in the group can exactly run the distance ‘x’. For example, if x is 7 and if there are six runners with capacity as 1 3 2 6 1 7 then players with capacity 1 3 2 and 1 with indices 1, 2, 3, 5 can form the team for relay race.

**Boundary conditions**

Only one such team may exist

**Input Format**

First line the value of x

Next line contains the value of n, number of people in the group

Next line contains the capacity of the n runners separated by a space

**Output Format**

Print the index of the players who can form the team so that the sum of their capacity exactly matches x. Index starts with 1

If no such team exist then print No

**ANS.  
from itertools import combinations**

**x,n,runner=int(input()),int(input()),list(map(int,input().split()))**

**comb=set(combinations(runner,4))**

**for i in comb:**

**if sum(i)==x:**

**for j in i:**

**print(runner.index(j)+1,end=" ")**

**runner[runner.index(j)]=-1**

**exit()**

**print("No")**

#### **Weigh the Vegetables**

A vegetable whole sale vendor got a request to supply ‘n’ vegetables each of quantity ‘x’ kg from the chef of a hotel. A lorry was loaded with the packets of ‘n’ vegetables each weighing ‘x’ kg and sacks of all ‘n’ vegetables to be preserved in the godown. They had two weighing stones of weight w1 and w2. When they reached the hotel a new list with different weights were given to the vendor. All new weights are greater than or equal to ‘x’. The vendor will deliver the packed vegetables. In addition, the excess quantity in the new list will be exactly delivered if they can be weighed with weighing stones w1 and w2. The vegetables are weighed using the weighing stones only and the weighing stones can be used many times. Given the value of x, w1, w2 and a set of new weights, write a C++ program to find the weight of vegetables that can be delivered.

For example, if x is 10, w1 is 3, w2 is 7 and the new weights are 19, 22, 21, 24 then the weight of vegetables that will be exactly delivered are 19, 22, 24.

**Input Format**

First line contains the value of x

Next line contains the value of w1

Next line contains the value of w2

Next line contains the value of n

Next line contains value of n values separated by a space

**Output Format**

Print 1 for weights that can be exactly weighed and 0 otherwise

**ANS.**

**x,w1,w2,n,wts=int(input()),int(input()),int(input()),int(input()),list(map(int,input().split()))**

**for i in wts:**

**if (i-x)%w1==0 or (i-x)%w2==0 or (i-x)%(w1+w2)==0: print(1,end=" ")**

**else: print(0,end=" ")**

**C++ CODE:**

**#include <iostream>**

**using namespace std;**

**int main(){**

**int x,w1,w2,n;**

**cin>>x>>w1>>w2>>n;**

**int arr[n];**

**for(int i=0;i<n;i++){cin>>arr[i];}**

**for(int i=0;i<n;i++){**

**if ((arr[i]-x)%w1==0 || (arr[i]-x)%w2==0 || (arr[i]-x)%(w2+w1)==0){**

**cout<<1<<" ";**

**}**

**else{cout<<0<<" ";}**

**}}**

#### **Is Cattle Safe**

A grazing field is in the shape of a rectangle. There is no physical fencing for the field. The field is bordered by yellow lines on all the four sides. If the cattle crosses the border on any side, the owner of the adjacent will hold the cattle as a captive. A farmer leaves his cattle inside the field in the morning and goes for his farm work. The farmer wants to develop a monitoring system so that farmer receives a beep alert in his mobile whenever the cattle is 1 cm nearer to the border on any side. The corners of the rectangle are designated as left-bottom, right-bottom, left-top, right-top. The left-bottom corner of the field is origin, in the 2-dimensional plane with x-axis and y-axis. All the coordinates of all the corners involve positive integers only. Breadth of the field is greater than the length of it. For example, if the diagonal of the field is 10 cm and the position of the cattle is (5, 3) then the cattle is in safe region.

Given the length of the diagonal of the field and the position of cattle (X,Y), write a code to identify whether the cattle is within a distance of 1 cm from the border on any side.

**Input Format**

First line contains the length of the diagonal

Next line contains the position of the cattle x and y values are separated by a space

**Output Format**

Print either Safe or Unsafe

**ANS.**

**diagonal = int(input())**

**Ax,Ay = map(float,input().split())**

**#left bottom corner of rectangle is the origin(O) and A(Ax,Ay) is our given point**

**slope\_of\_the\_line\_OA = Ay/Ax;**

**#BASE = PERPENDICULAR\*COT(THETA)**

**projection\_of\_line\_OA\_on\_Yaxis = Ax/slope\_of\_the\_line\_OA**

**#COT(THETA) FOR 45 DEGREE = 1 AND DIAGONAL MAKES 45 DEGREE WITH Y-AXIS**

**projection\_of\_diagonal\_on\_Yaxis = diagonal/1**

**if projection\_of\_line\_OA\_on\_Yaxis < projection\_of\_diagonal\_on\_Yaxis:**

**print("Safe")**

**else: print("Unsafe")**

#### **Longest Vertical Line**

Given ‘n’ points, write a C++ code to find the longest vertical line that can be formed and the number of points to be used to form the line. Two points can be joined to form a vertical line if their x-coordinates are same and y-coordinates are different. Length of the line is defined as the distance between the start and end point of the line. For example, given seven points (2, 11), (2, 3), (7, 8), (2, 5), (5, 50), (5, 6) and (7, 3) the longest vertical line can be obtained by joining point (5, 6) and (5, 50) of length 44.

**Input Format**

First line contains the number of points, n

Next n lines contains the coordinates of the points x and y separated by a space

**Output Format**

First line should contain the starting point of the longest vertical line

Next line should contain the ending point of the longest vertical line

Next line should contain the length of the longest vertical line

Next line should contain the number of points used to form the longest vertical line

**ANS.**

**n = int(input())**

**pts = [tuple(map(int,input().split())) for i in range(n)]**

**max\_d,max\_1,max\_2 = 0,tuple(),tuple()**

**for i in range(n):**

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

**if pts[i][0] == pts[j][0]:**

**d = abs(pts[i][1]-pts[j][1])**

**if max\_d < d:**

**max\_d = d**

**if pts[j][1] > pts[i][1]: max\_1,max\_2 = pts[i],pts[j]**

**else: max\_1,max\_2 = pts[j],pts[i]**

**print(\*max\_1)**

**print(\*max\_2)**

**count = 0**

**for i in pts:**

**if i[0] == max\_1[0]: count+=1**

**print(max\_d,count,sep="\n")**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**#include <cmath>**

**using namespace std;**

**int main(){**

**int n;**

**cin>>n;**

**vector <pair<int,int>> pts(n);**

**for (int i=0;i<n;i++) {**

**cin>>pts[i].first>>pts[i].second;**

**}**

**pair <int,int> max\_1,max\_2;**

**int max\_d = 0;**

**for (int i=0; i<n; i++) {**

**for(int j=i+1; j<n; j++) {**

**if (pts[i].first == pts[j].first) {**

**int d = abs( pts[i].second - pts[j].second );**

**if (max\_d < d) {**

**max\_d = d;**

**if (pts[j].second > pts[i].second) {**

**max\_1 = pts[i];**

**max\_2 = pts[j];**

**}**

**else {**

**max\_1 = pts[j];**

**max\_2 = pts[i];**

**}**

**}}}}**

**cout<<max\_1.first<<" "<<max\_1.second<<"\n";**

**cout<<max\_2.first<<" "<<max\_2.second<<"\n";**

**int count = 0;**

**for(int i=0;i<n;i++){**

**if (max\_1.first == pts[i].first) { count++; }**

**}**

**cout<<max\_d<<"\n"<<count;**

**}**

#### **Pair of Rabbits in Field**

A new born rabbit attains the capability to reproduce in two months and thereafter it reproduces every month. Here assume that a rabbit can reproduce a pair of rabbits every month, one of them is male and the other one is female and the rabbits never die. Then if there are initially ‘n’ pair of rabbits then

1. At the end of the first month there will be n pair of rabbits only

2. At the end of second month the female produces a new pair, so now there will be 2\*n pairs of rabbits in the field.

**3.**At the end of the third month, the original ‘n’ female produces a second pair, making 3\*n pairs in all in the field.

4. At the end of the fourth month, the original ‘n’ females has produced yet another ‘n’ pair, the females born two months ago produces their first pairs also, making 5\*n pairs and so on

A farm owner has given some money to his worker to buy a maximum of ‘n’ pairs of rabbits and told that if they maintain the farm well so that no rabbits die and the pair of rabbits at the end of ‘m’ month is divisible by 3 then he will be rewarded. The worker is confident to maintain the farm clean but doesn’t know how to select initial pair of rabbits so that he can be rewarded. Write a C++ code to find the maximum initial number. For example, if the value of n is 8 and m is 20 then the output will be 8 and 54120.

**Boundary Conditions**

m>=2

**Input Format**

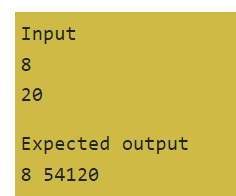
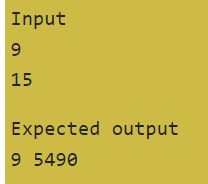
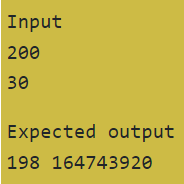
First line contains the value of n

Next line contains the value of m

**Output Format**

Print the number of pairs to be chosen initially and the number of pair of rabbits at the end of mth month separated by a space

**ANS. TEST CASES:**

**  **

**C++ CODE:**

**#include <iostream>**

**using namespace std;**

**#define ll long long**

**ll fib(ll m){**

**ll f=1,s=1,term;**

**for(ll i=0;i<m-2;i++){**

**term = f+s;**

**f = s;**

**s = term;**

**}**

**return term;**

**}**

**int main(){**

**ll n,m;**

**cin>>n>>m;**

**ll mul = fib(m);**

**while(n > 0){**

**ll ans = mul\*n;**

**if((mul\*n)%3 == 0){**

**cout<<n<<" "<<ans;**

**exit(0);**

**}**

**n--;**

**}**

**}**

#### **Power Related Numbers**

A number ‘x’ is a power related number if sum of digits of pow(x,p) = x where p is an integer >=2. For example, 8 is a power related number as sum of digits of pow(8, 3) = 8. Given the value of lower limit l, upper limit u and a number p, write a C++ code to find the minimum and maximum power realted numbers.

For example, if l is 1, u is 100, p is 5 then the minimum power related number is 1 with power as 5 and pow(1, 5) = 1 and maximum power related number is 46 with power as 5 and power (46, 5) = 205962976.

**Input Format**

First line contains the value of lower limit, l

Next line contains the value of upper limit, u

Next line contains the value of power, p

**Output Format**

First line should contain the minimum power related number, its power and pow(min, p) separated by a space

Next line should contain the maximum power related number, its power and pow(max, p) separated by a space

**ANS.**

**from math import pow**

**l,u,p = int(input()), int(input()), int(input())**

**ml = pl = powl = u**

**mu = pu = powu = l**

**for i in range(l,u+1):**

**for j in range(2,p+1):**

**if i == sum(list(map(int,list(str(int(pow(i,j))))))):**

**if i < ml: ml,pl,powl = i, j, int(pow(i,j))**

**if i > mu: mu,pu,powu = i, j, int(pow(i,j))**

**print(ml,pl,powl)**

**print(mu,pu,powu)**

**MY CODE CORRECT BUT VPROPEL DIDN’T ACCEPT:**

**from math import pow**

**l,u,p=int(input()),int(input()),int(input())**

**nums=[]**

**for i in range(l,u+1):**

**for k in range(2,p+1):**

**dig=sum(list(map(int,list(str(int(pow(i,k)))))))**

**if dig==i: nums.append((i,k,int(pow(i,k))))**

**print(\*min(nums))**

**print(\*max(nums))**

#### **New Number System**

A new k – base number system using the symbols 0, 1, 2, ..., k-1 is developed. Each number in the k-base number system is separated by a comma. Let a1,a2,a3 be a number in k-base number system where ai f ε {0, 1, 2, ..., k-1} then value of the number in decimal number system is a3\*k2+a2\*k1+a1\*k0. In a 18-based number system, value of the number 16, 17, 15 is 15\*182+17\*181+15\*180. Given two numbers in k-base number system, write a code to print the sum of the numbers in k-base system.

For example, if the two numbers n1 and n2 to base 9 are 6,8,7 and 7,8,0 then the sum of n1 and n2 are 4,8,8.

**Note:**

Use atoi for string to integer convesion. Refer the below code for reference

#include <iostream>

using namespace std;

int main ()

{

char s[100];

cin>>s;

int i=atoi(s);

cout<<i+10;

}

**Input Format**

First line contains the base of number system, k

Next line contains the number n1 in k-base number system

Next line contains the number n2 in k-base number system

**Output Format**

Print the sum of n1 and n2 in k-base number system

**ANS.**

**#code written acc to new base system in ques not the conventional system**

**#in conventional system reverse of the lists was to be used given**

**def convert(x,b):**

**k,dec=0,0**

**for i in x:**

**dec += int(i)\*(b\*\*k)**

**k += 1**

**return dec**

**def rev\_convert(x,b):**

**based=[]**

**while x != 0:**

**based.append(str(x%b))**

**x //= b**

**return ",".join(based)**

**n1,n2,k = input().split(',') , input().split(',') , int(input())**

**k1,k2 = convert(n1,k) , convert(n2,k)**

**print(rev\_convert(k1+k2,k))**

**C++ CODE:**

**#include <iostream>**

**#include <cmath>**

**#include <string>**

**#include <sstream>**

**#include <vector>**

**using namespace std;**

**//function - how to input a string word wise with delimiters such as comma and space etc.**

**//Returns a vector containing only the words of the string**

**vector <string> input (string s,char y){**

**vector <string> num;**

**stringstream s1(s);**

**while (s1.good()){**

**string word;**

**getline(s1,word,y);**

**num.push\_back(word);**

**}**

**return num;**

**}**

**int convert(vector <string> x,int b){**

**int k = x.size()-1,dec=0;**

**for(int i = x.size()-1 ; i>-1 ; i--){**

**dec += stoi(x[i])\*pow(b,k);**

**k--;**

**}**

**return dec;**

**}**

**void rev\_convert (int x,int b) {**

**while(x>0){**

**if(x/b>0){**

**cout<<x%b<<',';**

**x/=b;**

**}**

**else {**

**cout<<x%b;**

**break;**

**}}**

**}**

**int main(){**

**string x,y;**

**int k;**

**cin>>x>>y>>k;**

**rev\_convert( convert(input(x,','),k) + convert(input(y,','),k) , k );**

**}**

#### **Goto Competiton or Not**

A set of students of a college are getting prepared for an international coding competition ICPC conducted by ACM. A college has appointed a coach and they will permit a student to the competition only if he approves. The coach has formulated a rule for approving a student. He prepares ‘m’ number of schedules that consist of number of problems and a deadline. If a student has met atleast half of the deadline then he will be permitted to go for the competition (when the number of schedule is odd consider (m+1)/2 as half). Since he wanted the students work consistently he didn’t let the students to know the schedule. But students are lethargic and they solve some number of problems on a particular day and leaves a gap they are not working consistently each day.

For example when there are two schedules given by the coach on 9th day

the student must have completed 2 problems (9 2) and on 15th the student

must have completed 7 problems (15, 7) and the student has worked on

three days that is on 10th day he completes 5 problems, 14th day he

completes 4 problems and 31st day he completes 1 problem then the

student is allowed to participate the competition as he has met

deadline of one schedule out of two.

Given the detials of the schedule by the coach and number of problems solved by a student, write a C++ code to check if that student is allowed for the competition or not.

**Input format**

First line contains the number of schedules by the coach, m

Next ‘m’ lines contain the number of problems to be solved by the student and the deadline (day number) separated by a space

Next line contains the value of number of days the student works, n

Next ‘n’ lines contain the day number and the number of problems solved by the student separated by a space

**Output Format**

Print either Allowed or Denied

**ANS.**

**m=int(input())**

**deadlines=[tuple(map(int,input().split())) for i in range(m)]**

**stud\_exec=[tuple(map(int,input().split())) for i in range(int(input()))]**

**d\_completed,j,solved= 0,0,0**

**for i in deadlines:**

**while stud\_exec[j][0] < i[0]:**

**if j > len(stud\_exec)-2: break;**

**solved+=stud\_exec[j][1]**

**j+=1**

**if solved>=i[1]: d\_completed+=1**

**if m%2==0 and d\_completed>=m//2: print("Allowed"); exit()**

**if m%2!=0 and d\_completed>=(m+1)//2 : print("Allowed"); exit()**

**print("Denied")**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**#include <cstdlib>**

**using namespace std;**

**int main(){**

**int m,n;**

**cin>>m;**

**vector <pair<int,int>> deadlines(m);**

**for(int i=0;i<m;i++){**

**cin>>deadlines[i].first>>deadlines[i].second;**

**}**

**cin>>n;**

**vector <pair<int,int>> stud\_exec(n);**

**for(int i=0;i<n;i++){**

**cin>>stud\_exec[i].first>>stud\_exec[i].second;**

**}**

**int d\_completed=0,solved=0,j=0;**

**vector <pair<int,int>> :: iterator it;**

**for(it=deadlines.begin(); it!=deadlines.end(); it++){**

**while(stud\_exec[j].first < it->first){**

**if (j > n-2) { break; }**

**solved += stud\_exec[j].second;**

**j++;**

**}**

**if (solved >= it->second) { d\_completed++; }**

**}**

**if ( m%2 == 0 and d\_completed >= m/2 ) { cout<<"Allowed"; exit(0); }**

**if ( m%2 != 0 and d\_completed >= (m+1)/2 ) { cout<<"Allowed"; exit(0); }**

**cout<<"Denied";**

**}**

#### **Minimum Cost Laptop**

Ganesh has got some amount ‘x’ in hand and he wants to purchase a laptop of cost less than or equal to x. But he likes the laptop to have a bigger screen. Given the amount ‘x’ in hand and the details of laptop, write a C++ code to print the details of the laptop that fits in budget and has maximum screen area. If no laptop fits in his budget then print ‘No laptop’.

**Input Format**

First line contains the amount ‘x’ in hand

Next line contains the number of laptops, n

Next ‘n’ lines contain the details of the laptops – Length of screen, breadth of screen and cost of laptop

**Output Format**

Print the detail of laptop that fits in his budget and has maximum screen size and print ‘No laptop’ otherwise

**ANS.**

**x,n,info,info2=int(input()),int(input()),[],[]**

**for i in range(n):**

**det=tuple(map(int,input().split()))**

**info.append(det)**

**for i in info:**

**if i[2] <= x: info2.append(i)**

**try: print(\*max(info2,key = lambda a: a[2]))**

**except: print("No laptop")**

#### **Product of Two 2-Digit with ‘+’ and ‘/’**

Based on the yesterday’s problem told by Karthik, Vijay got curious to know more about tricks by Shakuntala Devi and he started reading her books. He liked the trick which describes a method to find product of two 2-digit numbers with ‘+’ and ‘/’ operators only. Given two 2-digit numbers ‘m’ and ‘n’,

(i) Write the lesser number ‘l’ on left side of a paper

(ii) Divide ‘l’ by 2 and write quotient on one line ignoring remainders till 1 is reached.

(iii) Write the larger number on the right side of the paper in line number 1

(iv) Then for right side of each line, write double of the value in previous line.

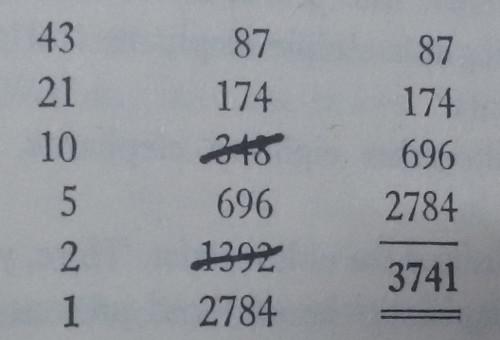
(v) Strike out the lines of even quotients in left side.

(vi) Sum up the remaining values on the right side

This gives product of m and n.

Given two 2-digit numbers, write a C++ program to find product of the numbers using above procedure and print the values on the left and right sides of the papers that are not striked out in ascending order of left side value.

For example, if the numbers are 43 and 87 then the procedure works as follows:

****

And the expected output is:

1 2784

5 696

21 174

43 87

3741

**Input Format**

First line contains the first two digit number, m

Next line contains the second two digit number, n

**Output Format**

Print all the lines that are not striked out in the above procedure

Print the product of m and n

**ANS.**

**m,n=int(input()),int(input())**

**x,y = min(m,n),max(m,n)**

**k,l=[],[]**

**while x>=1:**

**k.append(x)**

**x//=2**

**l.append(y)**

**y\*=2**

**k.reverse(); l.reverse()**

**for i in range(len(k)):**

**if k[i]%2==0: l[i] = -1**

**else:**

**print(k[i],l[i])**

**sum=0**

**for i in l:**

**if i==-1: pass**

**else: sum+=i**

**print(sum)**

**C++ CODE:**

**#include <iostream>**

**using namespace std;**

**#include <deque>**

**#define long long int**

**int32\_t main(){**

**int m,n,x,y;**

**cin>>m>>n;**

**if (m>n) {x=n; y=m;}**

**else {x=m; y=n;}**

**deque <int> k,l;**

**while(x>=1){**

**k.push\_front(x);**

**x/=2;**

**l.push\_front(y);**

**y\*=2;**

**}**

**for(int i=0;i<k.size();i++){**

**if(k[i]%2==0){**

**l[i] = -1;**

**}**

**else{**

**cout<<k[i]<<" "<<l[i]<<"\n";**

**}**

**}**

**int sum = 0;**

**deque <int> :: iterator it;**

**for(it=l.begin();it!=l.end();it++){**

**if (\*it == -1) {continue;}**

**else {sum += \*it;}**

**}**

**cout<<sum;**

**}**

#### **Nullify Bits**

For a given integer, odd\_bit\_value is defined as the number of bits which are '1' in odd position and even\_bit\_value is defined as the number of bits which are '1' in even position. Nullification is a process in which the 1's in odd position are made as zero if odd\_bit\_value is lesser than even\_bit\_value and it makes 1's in even position as zeros otherwise.

In this process, number of bits to be considered for this process is equal to the minimum number of bits required to represent the number. The right most bit of the number is considered to at poistion 1. For example, if n is 12 - 1100, it becomes 4 and 15 becomes 5.

Given an integer, write a C++ program to print the number after nullification.

**Input Format**

First line contains number, n

**Output Format**

Print the number after nullification

**ANS.**

**n=list(reversed(bin(int(input()))[2:]))**

**even\_bit\_value = 0**

**odd\_bit\_value = 0**

**for i in range(len(n)):**

**if (i+1)%2==0 and n[i]=='1': even\_bit\_value+=1**

**if (i+1)%2!=0 and n[i]=='1': odd\_bit\_value+=1**

**if odd\_bit\_value < even\_bit\_value:**

**for i in range(len(n)):**

**if (i+1)%2!=0 and n[i]=='1': n[i]='0'**

**else:**

**for i in range(len(n)):**

**if (i+1)%2==0 and n[i]=='1': n[i]='0'**

**print(int("".join(list(reversed(n))),2))**

#### **Ticket Number that Yields Maximum Points**

A college is conducting a series of cultural events by inviting celebrities. A group of student volunteers are involved in organising the events. For each event, a student wants to attend, he has to buy a ticket for the event by paying the price mentioned by the organisers. To encourage the participation of students the organisers have announced a type of discount. That is in each ticket for event E1, a random value of point ‘p’ is printed and there will be discount in the price of the ticket for event E2 based on ‘p’. Higher the value of ‘p’, higher the discount will be. When a group of ‘k’ friends come together to buy the tickets, they will be given continuous ‘k’ tickets and they can choose the starting ticket number.

Given the details of the tickets available and number of friends who have come to buy the tickets write a C++ code to determine the first starting ticket number which will yield maximum points for them.

For example, when there are totally nine tickets available and a group of three friends have come to buy tickets and the details of the tickets(ticket number and points) are as follows

10 1

11 2

12 3

13 1

14 4

15 5

16 2

17 3

18 7

They have to choose 16 as starting ticket number to yield maximum points.

**Input Format**

First line contains the total number of tickets available ‘n’ and the number of friends in the group ‘k’ separated by a space

Next ‘n’ lines details on the tickets (ticket number and points printed on the ticket) separated by a space

**Output Format**

Print the starting ticket number that will yield maximum points for the group of friends

**ANS.**

**n,k=map(int,input().split())**

**l=[]**

**for i in range(n):**

**t=tuple(map(int,input().split()))**

**l.append(t)**

**max,x=0,0**

**for i in range(n-k+1):**

**sum=0**

**for j in range(i,i+k):**

**sum+=l[j][1]**

**if sum>max:**

**max=sum**

**x=l[i][0]**

**print(x)**

#### **Wet Will (testament)**

A rich man of a city wrote a will (testament) to distribute his property to his nine children. In addition to lands, houses and jewels, he had also deposited some money in his bank account. The rich man passed away one day and the properties were distributed to his children as per his will. But a part of his will got wet in which one of the digit at some position 'p' of the money that he deposited in his bank account got erased. The rich man’s friend told that the amount (in rupees) deposited by the rich man was such that it can be evenly distributed (in rupees) to his nine children without leaving any remaining rupee(s). For example, if the will contains ₹13456 then the missing digit is 8.

Given the amount written in the will without the missing digit at some position 'p', write a C++ program to find the missing digit so that maximum amount that could have been written on the will.

**Input Format**

First line contains the amount without missing digit at some position 'p', n

**Output Format**

Print the missing digit

**ANS.**

**PYTHON CODE:**

**n=list(map(int,str(int(input()))))**

**x,k=sum(n),1**

**while((x+k)%9!=0): k+=1**

**print(k)**

**C++ CODE:**

**#include <iostream>**

**#define int long long**

**using namespace std;**

**int32\_t main(){**

**int n,d,sum=0,k=1;**

**cin>>n;**

**while(n!=0){**

**d=n%10;**

**sum+=d;**

**n/=10;**

**}**

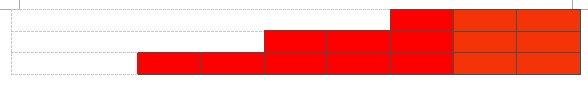
**while((sum+k)%9!=0){k++;}**

**cout<<k;**

**}**

#### **Build a Staircase**

Each step in a staircase built by a constructor has two bricks less than the number of bricks used in the previous step as shown below.

****

Given the number of bricks ‘n’ available in a construction site and the number of steps to be built ‘k’, write a C++ program to check if ‘k’ steps can be built with exactly ‘n’ bricks. If ‘k’ steps can be built then print the number of bricks to be used in each step starting from the last brick and print ‘Cannot be built’ otherwise.

For example, if ‘n’ is 15 and ‘k’ is 3 then print 3, 5 and 7 and if ‘n’ is 17 and ‘k’ is 3 then print ‘Cannot be built’.

**Input Format**

First line contains the number of bricks in the construction site, n

Next line contains the number of steps to be built, k

**Output Format**

Print the number of bricks to be used in each step, starting from the last step if ‘k’ steps can be built with exactly ‘n’ bricks and print ‘Cannot be built’ otherwise

**ANS.**

**PYTHON CODE:**

**n,k,d=int(input()),int(input()),2**

**if n/k == n//k: a=(n//k)-(k-1)\*(d//2)**

**else: print("Cannot be built"); exit()**

**for i in range(1,k+1):**

**print(a,end="\t")**

**a+=d**

**C++ CODE:**

**#include <iostream>**

**#define int long long**

**using namespace std;**

**int32\_t main(){**

**int n,k,d=2,a;**

**cin>>n>>k;**

**if(n%k==0){ a = (n/k)-(k-1)\*(d/2); }**

**else{**

**cout<<"Cannot be built";**

**exit(0);**

**}**

**for(int i=1;i<=k;i++){**

**cout<<a<<"\t";**

**a+=d;**

**}**

**}**

**NEW APPROACH (C++):**

**#include <iostream>**

**#include <vector>**

**#define int long long**

**using namespace std;**

**int32\_t main(){**

**int n,k; cin>>n>>k;**

**//problem is to find k consecutive odd nos. whose sum is n and I reverse engineered the equation**

**bool flag = false;**

**double first\_term = (double)(n-k\*(k-1))/k;**

**if(first\_term-(double)(int)first\_term == 0) flag = true;**

**if(flag){**

**int f\_t = (int)first\_term;**

**if(f\_t%2){ //HERE I CHECK WHETHER FIRST TERM IS ODD OR NOT**

**int i=0;**

**while(i < k){**

**cout<<f\_t<<"\t";**

**f\_t+=2;**

**i++;**

**}**

**}**

**else cout<<"Cannot be built";**

**}**

**else cout<<"Cannot be built";**

**}**

#### **Strength of Relation**

Strength of relation between two words S1 and S2 is defined by the set of position of common characters of S1 and S2, it is defined as Sum(pos(c) in S1 \* pos(c) in S2) where c is the character which is present in both S1 and S2. Given a collection of words S and a test string W, write a C++ program to find all strings in S with highest strength of relation with W. When there are many strings with highest strength of relation, print all strings in the order of input.

For example, consider the words thing and night, then the strength of relation (S1 to S2) is 1\*5+2\*4+3\*2+4\*1+5\*3 = 38.

Given a collection of five words S as apple, tiny, hit, thing and get and W as night, you should print thing which has the maximum strength of relation.

**Hint:**

This problem can be quickly solved with string in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use string, add #include<string>
* We can create object and array of objects for string
* length is a member function of string that gives the number of characters in it
* You can access ith character of a string object s as s[i] – Subscript operator is overloaded
* find is a member function that takes a character or string as argument and returns the position of occurrence if present and -1 otherwise

**Sample Code using String in STL**

#include<iostream>

using namespace std;

#include<string>

int main()

{

string s1,s2;

int l,pos;

cin>>s1;

cin>>s2;

//get length of s1

l=s1.length();

//check if first character of s1 is present in s2

pos=s2.find(s1[0]);

cout<<pos;

}

**Input Format**

First line contains the number of words ‘n’ in the collection, S

Next ‘n’ lines contain the words in S

Next line contains the word W

**Output Format**

Print all the words in S that has got maximum strength of relation

**ANS.**

**PYTHON CODE:**

**n=int(input())**

**words=[input() for i in range(n)]**

**w=input()**

**scores=[]**

**for j in words:**

**sum=0**

**for i in range(len(w)):**

**if w[i] in j:**

**sum+=i\*j.index(w[i])**

**scores.append((sum,j))**

**for i in scores:**

**if i[0]==max(scores)[0]: print(i[1])**

**C++ USING MAP:**

**#include <iostream>**

**#include <string>**

**#include <map>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**string words[n],w;**

**for(int i=0;i<n;i++) cin>>words[i];**

**cin>>w;**

**int max = 0;**

**map<int,int> info;**

**for(int i=0;i<n;i++){**

**int sum = 0;**

**for(int j=0;j<words[i].length();j++){**

**int pos = w.find(words[i][j]);**

**if(pos != -1) sum+=(j\*pos);**

**}**

**if(max < sum) max = sum;**

**info[i] = sum;**

**}**

**for(int i=0;i<n;i++){**

**if(info[i] == max) cout<<words[i]<<"\n";**

**}**

**}**

#### **Check Shuffled Sentence**

Given two sentences S1 and S2 formed with a set of distinct words, check if S1 is a shuffled sentence of S2. For example, if S1 = “this is a book” and S2 = “book is this a” then S1 is a shuffled sentence of S2.

**Hint**

This problem can be quickly solved with string in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use string, add #include<string>
* We can create object and array of objects for string
* To read a string s with spaces use getline(cin, s);
* A string object can be intialized to empty as string s="";
* length is a member function of string that gives the number of characters in it
* You can access ith character of a string object s as s[i] – Subscript operator is overloaded
* A character ‘c’ can be added to a string ‘s’ by s=s+c;

**Sample Code**

int main()

{

string sen1;

sen1="";

sen1=sen1+'a';

getline(cin,sen1);

int len=sen1.length();

}

**Input Format**

First line contains the sentence, S1

Next line contains the sentence, S2

**Output Format**

Print Yes if S1 is shuffled a sentence of S2 and No otherwise

**ANS.**

**PYTHON CODE:**

**s1,s2=sorted(list(input())),sorted(list(input()))**

**if "".join(s1)=="".join(s2): print("Yes")**

**else: print("No")**

**C++ CODE:**

**#include <iostream>**

**#include <string>**

**#include <algorithm>**

**using namespace std;**

**//how to remove whitespaces from string in C++**

**string s\_remove(string s){**

**for(int i=0;i<s.length();i++){**

**if(s[i]==32){s[i]='\0';}**

**}**

**return s;**

**}**

**int main(){**

**string s1,s2;**

**getline(cin,s1);**

**getline(cin,s2);**

**s1=s\_remove(s1);**

**s2=s\_remove(s2);**

**sort(s1.begin(),s1.end()); // how to sort a string in alphabetical order**

**sort(s2.begin(),s2.end());**

**if(s1.compare(s2)!=0){cout<<"No";} //how to compare two strings**

**else {cout<<"Yes";}**

**}**

**C++ new:**

**#include <iostream>**

**#include <string.h>**

**#include <algorithm>**

**#include <vector>**

**using namespace std;**

**vector<string> wordarr(char s[]){**

**vector<string> words;**

**string word = "";**

**for(int i=0;i<=strlen(s);i++){**

**if(s[i] == ' ' || s[i] == '\0'){**

**words.push\_back(word);**

**word = "";**

**}**

**word+=s[i];**

**}**

**return words;**

**}**

**int main(){**

**char s1[100],s2[100];**

**scanf("%[^\n]%\*c",s1);**

**scanf("%[^\n]%\*c",s2);**

**if(strlen(s1) != strlen(s2)) cout<<"No";**

**else{**

**vector<string> w1 = wordarr(s1),w2 = wordarr(s2);**

**for(auto x: w1){**

**if(find(w2.begin(),w2.end(),x) == w2.end()){**

**cout<<"No";**

**exit(0);**

**}**

**}**

**cout<<"Yes";**

**}**

**}**

#### **Lexicographically Ordered Sentences**

A sentence is said to be lexicographically ordered if the words of the sentence is either in lexicographically ascending or lexicographically descending order. For example, “apple is red” is in lexicographically ascending order and “zebra is animal” is in lexicographically descending order. Given ‘n’ sentences, write a C++ program to print lexicographically ordered sentences.

**Hint**

**When cin and getline are mixed, we have to clear the buffer using ignore function.**

**For example,**

**cin>>n;**

**cin.ignore();**

**getline(cin,s);**

This problem can be quickly solved with string in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use string, add #include<string>
* We can create object and array of objects for string
* To read a string s with spaces use getline(cin, s);
* A string object can be intialized to empty as string s="";
* length is a member function of string that gives the number of characters in it
* You can access ith character of a string object s as s[i] – Subscript operator is overloaded
* A character ‘c’ can be added to a string ‘s’ by s=s+c;
* Two strings s1 and s2 can be compared using <, > and ==

**Sample Code**

int main()

{

string sen1;

string w1,w2;

sen1="";

sen1=sen1+'a';

getline(cin,sen1);

cin>>w1>>w2;

int len=sen1.length();

if(w1<w2)

cout<<"W1 is lexicographically less than w2";

}

**Input Format**

First line contains the number of sentences, n

Next ‘n’ lines contain the sentences

**Output Format**

Print the sentences that is in some lexicographical order

**ANS.**

**PYTHON CODE:**

**sentences = [input().split() for i in range(int(input()))]**

**for i in sentences:**

**a=" ".join(i)**

**b=" ".join(sorted(i))**

**c=" ".join(sorted(i,reverse=True))**

**if a==b or a==c: print(a)**

**C++ CODE:**

**#include <iostream>**

**#include <string>**

**#include <vector>**

**using namespace std;**

**//self constructed strcmp function:**

**// int priority(string s1,string s2){**

**// int trav\_len;**

**// int n1 = s1.length(),n2 = s2.length();**

**// if(n1 >= n2) trav\_len = n2;**

**// else trav\_len = n1;**

**// for(int i=0;i<trav\_len;i++){**

**// if(s1[i] < s2[i]) return 1;**

**// if(s1[i] > s2[i]) return 2;**

**// }**

**// if(n1 > n2) return 2;**

**// else if(n1 == n2) return 0;**

**// else return 1;**

**// }**

**int main(){**

**int n; cin>>n;**

**cin.ignore();**

**for(int i=0;i<n;i++){**

**string sen;**

**getline(cin,sen);**

**vector<string> words;**

**string word="";**

**for(int j=0;j<=sen.length();j++){**

**if(sen[j] == ' ' || sen[j] == '\0'){**

**words.push\_back(word);**

**word = "";**

**}**

**else word+=sen[j];**

**}**

**int flag = 0,x=words.size();**

**for(int k=0;k<x-1;k++){**

**if(words[k] <= words[k+1]){**

**flag = 1;**

**break;**

**}**

**}**

**if(flag == 0) cout<<sen<<"\n";**

**else{**

**for(int k=x-1;k>=1;k--){**

**if(words[k] <= words[k-1]){**

**flag = 2;**

**break;**

**}**

**}**

**if(flag == 1) cout<<sen<<"\n";**

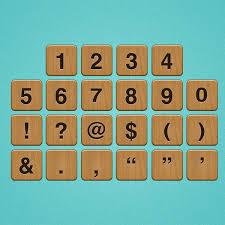
**}**

**}**

**}**

#### **Joined Number**

A set of house numbers were formed using number tiles as shown below. During transportation, the numbers got joined. Vivek knows the sum of digits of his house number but do not know the exact house number and the number of digits in it.

****

Given the joined number ‘n’ and the sum of house number ‘h’, write a C++ program to find all possible consequent digits of ‘n’ whose sum of digits is ‘h’. For example, if the joined number ‘n’ is 7234198523322 and house number ‘h’ as 17, the answers are 72341, 3419, 98 and 523322

**Hint:**

This problem can be quickly solved with string in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use string, add #include<string>
* We can create object and array of objects for string
* length is a member function of string that gives the number of characters in it
* You can access ith character of a string object s as s[i] – Subscript operator is overloaded
* substr(i,n) – will return a substring from position ‘i’ and of length ‘n’
* stol(s) – returns long integer of string s

**Sample Code using String in STL**

#include<iostream>

using namespace std;

#include<string>

int main()

{

string s1,s2;

int l,pos;

long int li;

cin>>s1;

//get length of s1

l=s1.length();

//substring of length 3 from position 2

s2 = s1.substr(2,3);

li = stol(s1);

**}**

**Input Format**

First line contains the joined number, n

Next line contains the sum of the house number, h

**Output Format**

Print the consequent digits that may be house number separated by a tab

**ANS.**

**n,h=input(),int(input())**

**x=len(n)**

**for i in range(x):**

**for j in range(i+1,x+1):**

**sum=0**

**for k in n[i:j]:**

**sum+=int(k)**

**if sum==h: print(n[i:j],end="\t")**

**C++ CODE:**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**int main(){**

**string n;**

**getline(cin,n);**

**int h,l;**

**cin>>h;**

**l=n.length();**

**for(int i=0;i<l;i++){**

**for(int j=i+1;j<l+1;j++){**

**int sum=0;**

**string y=n.substr(i,j-i);**

**for(int k=0;k<y.length();k++){**

**sum+=((int)(y[k])-48);**

**}**

**if (sum==h){cout<<y<<"\t";}**

**}**

**}**

**}**

#### **Swap Details Entered**

An web application was designed by a company to calculate the income tax of its employees as per the latest budget. A form was designed to enter the details of the employees and do income tax calculation. Since the basic excemption limit, percentage of deductions etc differ based on gender, separate forms are designed in the web application for male and female. Number of fields to be entered and width of the fields to be entered are same. By mistake both the genders have entered details in the form for the other gender. Internally the details entered by each employee is stored as a string with a delimiter in between. For males ‘:’ is used as the delimiter and for females ‘#’ is used as delimiter.

Given the strings of details of a male and a female employee, write a C++ code to swap the corressponding fields and make it ready for further processing.

For example, if the strings given are 50891:12378949:10000:ACZPJ9823B and 78191#45376107#10200#BFZPJ0453B then the output should be 78191:45376107:10200:BFZPJ0453B and 50891#12378949#10000#ACZPJ9823B

**Hint:**

This problem can be quickly solved with string in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use string, add #include<string>
* We can create object and array of objects for string
* length is a member function of string that gives the number of characters in it
* find(s, pos) – will return the position of the string s from position pos and return -1 if cannot be found
* substr(i,n) – will return a substring from position ‘i’ and of length ‘n’
* replace (p1,p2,s) – replaces part of the string from p1 to p2 by s

**Sample Code using String in STL**

#include<iostream>

using namespace std;

#include<string>

int main()

{

string s1,s2;

int l,pos;

cin>>s1;

//get length of s1

l=s1.length();

//substring of length 3 from position 2

s2 = s1.substr(2,3);

//searched for string s2 in s1 from position 3

l = s1.find(s2,3);

//replace part of string three characters from position 4 by s2

s1.replace(4,3,s2);

**}**

**Input Format**

First line contains the details entered by female with wrong delimiter

Next line contains the details entered by male with wrong delimiter

**Output Format**

First line contains the details entered by male with correct delimiter

**ANS.**

**d1,d2=input().split(":"),input().split("#")**

**print(":".join(d2),"#".join(d1),sep="\n")**

**C++ CODE:**

**#include <iostream>**

**#include <string.h>**

**using namespace std;**

**int main(){**

**char female[100],male[100];**

**scanf("%[^\n]%\*c",female);**

**scanf("\n");**

**scanf("%[^\n]%\*c",male);**

**for(int i=0;i<strlen(female);i++)**

**if(female[i] == ':') female[i] = '#';**

**for(int i=0;i<strlen(male);i++)**

**if(male[i] == '#') male[i] = ':';**

**printf("%s\n%s",male,female);**

**}**

#### **Missed Data and Average**

A mobile company collected data about usage of their mobile phones in various cities of the country. The details were then arranged such that the distance between the head office and the city is increasing. While data was entered they missed a data of a city. Given the data entered, missed data position and missed data, write a C++ program to print the data after insertion and average usage of every three adjacent cities.

For example, if six data are entered 13, 16, 19, 31, 45 and 67, missed data position is 2 and missed data is 50 then the data after insertion is 13, 50, 16, 19, 31, 45, 67 and average of three adjacent cities are 26.33, 28.33, 22.00, 31.67 and 47.67.

**Hint**

This problem can be quickly solved with vector in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use vector, add #include<vector>
* We can create object and array of objects for vector
* size() - Number of elements in a vector can be found
* [] - can be used to random access elements of a vector
* begin() - will give reference to the first element in the vector which can be stored in an iterator
* end() - will give reference to one position next to last element in the vector which can be stored in an iterator
* iterator for a integer vector with integer elements can be declared as vector<int>::iterator it
* +, -, ++, --, <, >, ==, != - operators that can be used with iterators
* insert(pos, e) – Inserts the element ‘e’ at the given iterator position pos

**Sample code**

#include<iostream>

using namespace std;

#include<vector>

int main()

{

vector<int> v;

v.push\_back(11);

v.push\_back(12);

vector<int>::iterator =v.begin();

//to insert data at 5th position

v1.insert(it+4,mis\_data);

for(int i=0;i<v.size();i++)

{

cout<<v[i]<<endl;

}

}

**Input Format**

First few lines contain the data entered and it is ended by -1

Next to the line -1 is the missed postion

Next line contains the missed data

**Output Format**

First line contains data after inserting missing data separated by a tab

Next line contains average of three cities with precision of two decimal place and separated by a tab

To print only two decimal places of variable f use:

#include<iomanip>

cout<<fixed<<setprecision(2)<<f;

**ANS.**

**data=[]; x=0**

**while(x!=-1):**

**x=int(input())**

**data.append(x)**

**data.pop(-1)**

**pos=int(input())-1**

**data.insert(pos,int(input()))**

**for i in range(len(data)):**

**print(data[i],end="\t")**

**print()**

**for i in range(len(data)-2):**

**print(format((data[i]+data[i+1]+data[i+2])/3,".2f"),end="\t")**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main(){**

**vector<int> data;**

**int ele,pos,val,ind=0;**

**while(true){**

**cin>>ele;**

**if(ele == -1) break;**

**data.push\_back(ele);**

**}**

**cin>>pos>>val;**

**data.insert(data.begin()+pos-1,val);**

**for(auto x: data) cout<<x<<"\t";**

**cout<<"\n";**

**for(int i=0;i<data.size()-2;i++){**

**printf("%.2f\t",(float)(data[i]+data[i+1]+data[i+2])/3);**

**}**

**}**

#### **Correlation Between Sequences**

There are two sequence generators G1 and G2 which generates floating point numbers. Both the generators have a small flaw and generates ‘n+1’ numbers when ‘n’ numbers need to be generated. Two sequences are said to be ‘Equal’ when the elements generated by them are same and the sequences are said to be ‘Good’ if there is a constant difference between the corresponding elements and the sequences are ‘Bad’ when it is not Equal and not Good.

Given two sequences generated by G1 and G2 and the positions of the erroneous elements, write a C++ code to remove the erroneous elements and check if the sequences are Equal, Good or Bad.

For example, if the sequences are 11.1, 12.5. 15.6, 17.01, 89.21, 35.6, 55.71 and 11.1, 13.45, 12.5, 15.6, 17.01, 35.6, 55.71 and the erroneous positions are 5 and 2 then the correct sequence of elements generated are 11.1, 12.5, 15.6, 17.01, 35.6, 55.71 and 11.1, 12.5, 15.6, 17.01, 35.6, 55.71, the sequences are ‘Equal’.

**Hint**

This problem can be quickly solved with vector in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use vector, add #include<vector>
* We can create object and array of objects for vector
* size() - Number of elements in a vector can be found
* [] - can be used to random access elements of a vector
* == - Returns true if the elements of two vectors are same and false otherwise
* begin() - will give reference to the first element in the vector which can be stored in an iterator
* end() - will give reference to one position next to last element in the vector which can be stored in an iterator
* iterator for a integer vector can be declared as vector<int>::iterator it
* +, -, ++, --, <, >, ==, != - operators that can be used with iterators
* erase(pos) – Removes the element at the given iterator position pos

**Sample code**

#include<iostream>

using namespace std;

#include<vector>

int main()

{

vector<int> v;

v.push\_back(11);

v.push\_back(12);

vector<int>::iterator it =v.begin();

//to delete data at 5th position

v1.erase(it+4);

for(int i=0;i<v.size();i++)

{

cout<<v[i]<<endl;

}

//to check if elements of two vectors are same

if(v1==v)

cout<<"Same";

}

**Input Format**

First line contains the number of elements generated by the sequence generators, n+1

Next line contains the elements generated by G1 separated by a space

Next line contains the elements generated by G2 separated by a space

Next line contains the position of erroneous element in the sequence generated by G1

Next line contains the position of erroneous element in the sequence generated by G2

**Output Format**

First line should contain elements generated by G1 after removing erroneous element

Next line should contain elements generated by G2 after removing erroneous element

Print if the sequences are Equal or Good or Bad

**Note:**In the expected output there is a space at the end of the sequences

**ANS.**

**C++ CODE:**

**#include <iostream>**

**#include <cmath>**

**#include <vector>**

**#define rep(i,a,b) for(int i=a;i<b;i++)**

**#define in(x) cin>>x**

**#define vf vector<float>**

**#define pb(v,x) v.push\_back(x)**

**using namespace std;**

**int main(){**

**int n; in(n);**

**vf g1,g2;**

**rep(i,0,n){**

**float x; in(x);**

**pb(g1,x);**

**}**

**rep(i,0,n){**

**float x; in(x);**

**pb(g2,x);**

**}**

**int pos1,pos2; in(pos1); in(pos2);**

**g1.erase(g1.begin()+pos1-1);**

**g2.erase(g2.begin()+pos2-1);**

**bool flag\_eq=true,flag\_c\_d=true;**

**rep(i,0,n){**

**if(flag\_eq == true){**

**if(g1[i] == g2[i]) continue;**

**else flag\_eq = false;**

**}**

**else{**

**if(i>0 && (g1[i-1]-g1[i-1])-(g1[i]-g2[i]) == 0) { //compare float values like this**

**flag\_c\_d = false; break;**

**}**

**}**

**}**

**for(auto x: g1) cout<<x<<" ";**

**cout<<"\n";**

**for(auto y: g2) cout<<y<<" ";**

**cout<<"\n";**

**if(flag\_eq) cout<<"Equal";**

**else if(flag\_c\_d) cout<<"Good";**

**else cout<<"Bad";**

**}**

#### **Periods in Periodic Table**

In a periodic table, all of the elements in a period have the same number of atomic orbitals. For example, every element in the top row (the first period) has one orbital for its electrons. Now the child we dealt in yesterday’s problem has grown up and he is in eigth class. His school wants him to remeber atleast five elements (except for the first period where there are only two elements) in each period of the periodic table. Given the period number, two elements in the first period and five elements in other six periods, write a C++ program to read a period number and print the elements under that particular period number.

**Hint**

This problem can be quickly solved with map in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use map, add #include<map>
* We have to give the data type of key and value during object creation. For example, map<int, int> m will create a map that can store an integer key and an integer value.
* Map can also have key and values of objects of classes such as string, vector etc.
* Value can be inserted into the map m as m[key] = value
* To check if a key is present in a map, use count function. It returns 0 – if key is not present and 1 – otherwise
* To take the value of an existing key in the map use value = m[key]

**Input Format**

First line contains the name of the two elements under period 1 in periodic table

Next line contains the name of the five elements under period 2 in periodic table

Next line contains the name of the five elements under period 3 in periodic table

Next line contains the name of the five elements under period 4 in periodic table

Next line contains the name of the five elements under period 5 in periodic table

Next line contains the name of the five elements under period 6 in periodic table

Next line contains the name of the five elements under period 7 in periodic table

Next line contains the period number, n

**Output Format**

Print name of the elements under period number n separated by a space

**ANS.**

**l=[]**

**for i in range(7):**

**l.append(tuple(map(str,input().split())))**

**n=int(input())**

**print(\*l[n-1])**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main(){**

**vector<vector<string>> table(7);**

**int n = 0,a;**

**while(n < 7){**

**if(n == 0) a = 2;**

**else a = 5;**

**for(int i=0;i<a;i++){**

**string ele; cin>>ele;**

**table[n].push\_back(ele);**

**}**

**n++;**

**}**

**int p; cin>>p;**

**for(auto x: table[p-1]) cout<<x<<" ";**

**}**

#### **Detailed Periodic Table**

The school wants the child to know few more details about five elements in each period of periodic table. So they ask the child to learn symbol, group, period, block, atomic number and state of the element at 20°**C** . Given the details of the five elements in each period (except period 1 – only two elements), period number and symbol of the desired element, write a C++ program to print the other details of the desired element.

**Hint**

This problem can be quickly solved with map in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* Using map of maps (that is map within map) will lead to simple code
* To use map, add #include<map>
* We have to give the data type of key and value during object creation. For example, map<int, int> m will create a map that can store an integer key and an integer value.
* Map can also have key and values of objects of classes such as string, vector etc.
* It is straightforward to have user defined data type as value of a map and we can also have a vector of userdefined datatypes as values
* A map can also have another map as value
* Value can be inserted into the map m as m[key] = value
* Value can also be inserted into a map using insert function, for example to insert a key k and value v into a map m, we can write m.insert({k, v}) where curly braces around k and v creates a pair to insert into the map
* To check if a key is present in a map, use count function. It returns 0 – if key is not present and 1 – otherwise
* To take the value of an existing key in the map use value = m[key]

**Input Format**

Details of elements is given in the following order - Name, Symbol, group, block, atomic number and state of the element at 20 °**C**

First two lines contain the details of the five elements in period 1

Next five lines contain the details of the five elements in period 2

Next five lines contain the details of the five elements in period 3

Next five lines contain the details of the five elements in period 4

Next five lines contain the details of the five elements in period 5

Next five lines contain the details of the five elements in period 6

Next five lines contain the details of the five elements in period 7

Next line contains the period of the desired element

Next line contains the symbol of the desired element

**Output Format**

Print name, group, block, atomic number and state of element separated by a space

**ANS.**

**elements=[]**

**for i in range(32):**

**elements.append(list(input().split()))**

**period\_no,symbol=int(input()),input()**

**table={1:elements[:2],2:elements[2:7],3:elements[7:12],4:elements[12:17],5:elements[17:22],6:elements[22:27],7:elements[27:32]}**

**for i in table[period\_no]:**

**if symbol == i[1]: i.pop(1); print(\*i)**

**C++ CODE:**

**#include <iostream>**

**#include <map>**

**#include <vector>**

**using namespace std;**

**int main(){**

**int n=7;**

**map<string,vector<string>> table;**

**while(n>0){**

**int a;**

**if(n==7) a=2;**

**else a=5;**

**for(int i=0;i<a;i++){**

**vector<string> ele(6);**

**for(int i=0;i<6;i++) cin>>ele[i];**

**string key = ele[1];**

**ele.erase(ele.begin()+1);**

**table[key] = ele;**

**}**

**n--;**

**}**

**int p; cin>>p; string sym; cin>>sym;**

**for(auto x: table[sym]) cout<<x<<" ";**

**}**

#### **Row Operations in a Matrix**

Four row operations are defined in matrix as follows:

1. Add a row (i, c) – where i is the index where insertion should happen and c is the collection of elements that has to be inserted (Index of row of matrix starts from 1).

2. Remove (i) - where i is the index where removal should happen

3. AddUp(i) – Returns sum of elements in the ith row

4. Even(i) – Returns the count of number of even elements in the ith row

This problem can be quickly solved with vector in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* Matrix can be viewed as vector of vectors
* To use vector, add #include<vector>
* We can create object and array of objects for vector
* size() - Number of elements in a vector can be found
* end() - will give reference to one position next to last element in the vector which can be stored in an iterator
* iterator for a integer vector can be declared as vector<int>::iterator it
* insert(pos, e) – Inserts the element ‘e’ at the given iterator position pos
* erase(pos) – erases the element at vector position pos

Algorithms of STL that can be used are:

* #include<algorithm> for STL algorithms
* counif(i, j, fn) – Function that counts the number of elements from iterator position i to iterator position j when function fn returns true

**Input Format**

First line contains the number of rows and columns in matrix, m

Next ‘r’ lines contain the elements of the ith row

Next line contains the value of ‘c’, choice of operation

Next line contains the value of ‘i’ for the operation

For choice 1, next row contains the elements of the new row that is to be inserted

**Output Format**

Print the matrix for choice 1 and 2 and print the value returned by functions for choice 3 and 4

While printing matrix, print one row in a line with a space between elements of a row

**Note:**There is a space at the end of each row

**ANS.**

**def print\_matrix(l):**

**for i in range(len(l)):**

**print(\*l[i],"")**

**r1,c1=map(int,input().split())**

**matrix = []**

**for i1 in range(r1):**

**row=list(map(int,input().split()))**

**matrix.append(row)**

**c=int(input())**

**i=int(input())**

**if c==1:**

**new\_row=list(map(int,input().split()))**

**matrix.insert(i-1,new\_row)**

**print\_matrix(matrix)**

**if c==2:**

**matrix.pop(i-1)**

**print\_matrix(matrix)**

**if c==3:**

**print(sum(matrix[i-1]))**

**if c==4:**

**count=0**

**for j in matrix[i-1]:**

**if j%2==0: count+=1**

**print(count)**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**void printmatrix(vector<vector<int>> m){**

**for(auto x: m){**

**for(auto y: x) cout<<y<<" ";**

**cout<<"\n";**

**}**

**}**

**int main(){**

**int r,c; cin>>r>>c;**

**vector<vector<int>> m;**

**for(int i=0;i<r;i++){**

**vector<int> vals;**

**for(int j=0;j<c;j++){**

**int ele; cin>>ele;**

**vals.push\_back(ele);**

**}**

**m.push\_back(vals);**

**}**

**int ch,i; cin>>ch>>i;**

**switch(ch){**

**case 1:{**

**vector<int> val(c);**

**for(int j=0;j<c;j++) cin>>val[j];**

**m.insert(m.begin()+i-1,val);**

**printmatrix(m);**

**break;**

**}**

**case 2:{**

**m.erase(m.begin()+i-1);**

**printmatrix(m);**

**break;**

**}**

**case 3:{**

**int sum=0;**

**for(auto x: m[i-1]) sum+=x;**

**cout<<sum;**

**break;**

**}**

**case 4:{**

**int even=0;**

**for(auto x: m[i-1])**

**if(!(x&1)) even++;**

**cout<<even;**

**break;**

**}**

**default: break;**

**}**

**}**

#### **Pick ith Ball after Arranging**

**smileyVPropel Celerates its 125th PoD for this Semester Todaysmiley**

Given the radius and color of ‘n’ balls and an index ‘i’, write a C++ program to arrange them in ascending order as per their volume and when volumes of two balls are same arrange them as per their name. After arranging, print the details of the ith ball. For example, when five balls are given with radius and color as below:

2 red

1 blue

1 red

2 blue

3 red

and index is given as 1, then after arranging in ascending order the balls will be:

1 blue

1 red

2 blue

2 red

3 red

therefore print 1 blue.

**Note:**Input can be very large

**Hint:**

This problem can be quickly solved using list in STL.

1. Design a class for ball overload < and > operator to decide about the realtionship between two balls

2. #include<list> includes list

3. list<int> l is used to create an integer list

4. l.sort() - will sort the list, provided < and > operators are overloaded for user defined data types

5. Random access is not possible in list

6. list<int>::iterator it=l.begin(); - creates an iterator for integer list and initializes to first element

7. Iterator can be moved by it++

8. value of an element by iterator it can be retrieved as \*it

**Input Format**

First line contains the number of balls, n

Next ‘n’ lines contain the details of the balls radius and color separated by a space

Next line contains the value of ‘i’

**Output Format**

Print the radius and color of ith ball after arrangement

**ANS.**

**n,details=int(input()),[]**

**for j in range(n):**

**dets=input().split()**

**dets[0]=int(dets[0])**

**details.append(dets)**

**i=int(input())**

**details.sort()**

**print(\*details[i-1])**

**C++ CODE:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**vector<pair<int,string>> det;**

**for(int i=0;i<n;i++){**

**int r; string color; cin>>r>>color;**

**pair<int,string> x = make\_pair(r,color);**

**det.push\_back(x);**

**}**

**int i; cin>>i;**

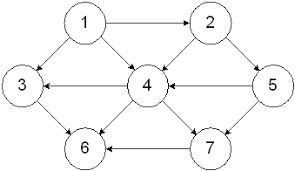
**sort(det.begin(),det.end());**

**cout<<det[i-1].first<<" "<<det[i-1].second;**

**}**

**Check for Flights**

In a country with ‘n’ cities, flight services are provided from some cities to certain cities only. The presence of flight service is represented by a directed graph as shown below. According to this graph, there are three flight services from city 1 to city 2, city 3 and city 4. There is no flight service from city 6.

****

Given the flight services available in a country and a source city ‘s’ and a destination city ‘d’, write a C++ code to check if a person from city ‘s’ can reach city ‘d’ using the flight services. If the person can reach then print 'Yes' and Print ‘No’ otherwise.

For example, if ‘s’ is 1 and ‘d’ is 6 then print Yes and if ‘s’ is 5 and d is 1 then print No

**Note:**

The graph formed by the flight services is acyclic

**Hint**

One of the idea to solve the problem is using the concept of Breadth First Search (BFS) of graphs. The steps involved using a queue are as follows:

1. Put the start city, s into a queue q

2. Repeat step 3 till either the queue becomes empty or destination city is found

3. Remove a city ‘c’ from q, if ‘c’ is ‘d’ then print Yes and stop otherwise find all cities that can be reached from c and put it inside the queue

Vectors, map and queue of STL can be used to quickly solve this problem. Queue is a data structure in which the element that is inserted first into the queue is removed first.

To create a queue in STL #include<queue> and give as queue<int> q; to create a queue of integers

Operations in queue of STL are

1. Push (e) – insert e into a queue at the end

2. front() - returns the first element in the queue

3. pop() - Remove the element in the front of the queue

4. Empty – Return true or false indicating if the queue is empty or not

**Input Format**

First line contains the number of cities in the country, n

Next ‘n’ lines contain the detail of flights to cities, each field in a line is separated by a space. First field is the city name ‘c’, next is the number of cities to which flight service is present from ‘c’, followed by the city names to which flight service is from ‘c’

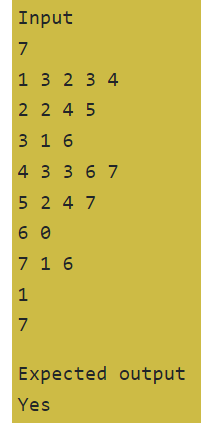
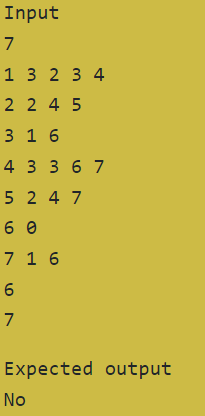
Next line contain the source city name from which the person wants to travel, s

Next line contain the destination city name to which the person wants to travel, d

**Output Format**

Print Yes if there exists connecting flights between s and d and No otherwise

**ANS. TEST CASES:**

** **

**ANS. YET TO BE FOUND.**

#### **Sum of First Few Consecutive Prime Numbers**

Given a number 'n', write an algorithm and a code to check if it can be written as sum of first few prime numbers. Print Yes if 'n' can be written as first few prime numbers and No otherwise. For example, if n is 5 then print Yes as it can be written as 2+3, if n is 41 then print Yes as it can be written as 2 + 3 + 5 + 7 + 11 + 13 and if n is 11 then print 'No' as it cannot be written as sum of first few prime numbers.

**Note: C++ compilers can compile C code also**

**Input Format**

First line contains the number, n

**Output Format**

First line contains Yes if the number n can be written as sum of first few prime numbers and No otherwise

**ANS.**

**JAVA**

**import java.util.\*;**

**import java.lang.\*;**

**class Main{**

**static boolean check\_prime(int n){**

**if(n<=1) return false;**

**if(n==2) return true;**

**if(n%2==0) return false;**

**for(int i=3;i<=Math.sqrt(n);i+=2)**

**if(n%i==0) return false;**

**return true;**

**}**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**int n = sc.nextInt();**

**int i=0,sum=0;**

**while(sum<n){**

**if(check\_prime(i) == true) sum+=i;**

**if(sum==n){**

**System.out.println("Yes");**

**System.exit(0);**

**}**

**i++;**

**}**

**System.out.println("No");**

**}**

**}**

**C++**

**#include <iostream>**

**#include <cmath>**

**#include <cstdlib>**

**using namespace std;**

**bool check\_prime(int n){**

**if(n<=1) return false;**

**if(n==2) return true;**

**if(n%2==0) return false;**

**for(int i=3;i<=sqrt(n);i+=2)**

**if(n%i==0) return false;**

**return true;**

**}**

**int main(){**

**int n,i=0,sum=0;**

**cin>>n;**

**while(sum<n){**

**if(check\_prime(i)==true) sum+=i;**

**if(sum==n){**

**cout<<"Yes";**

**exit(0);**

**}**

**i++;**

**}**

**cout<<"No";**

**}**

#### **Difference of the sum of alternate digits (D)**

Given a positive integer 'x' (with even number of digits in it), write an algorithm and the subsequent code to compute the difference between  the sum of the digits occuring in the alternate positions (starting from the first position) and the sum of the digits occuring in the alternate positions,starting from the last rightmost position of 'x'

For example, consider the number  8975.  The sum of the digits that occur in the alternate positions from the first position is 8+7=15.  The sum of the digits that occur in the alternate positions, starting from the rightmost position is 5+9 = 14. Difference between the two sums is 1 (=15-14).  Similarly, for the number 5798, the difference between  two sums, is 1.

**Note: Read the input as a number and do entire processing as  a number**

**C++ compilers can compile C code also**

**Input format**

First line contains the positive integer

**Output format :**

First line should contain the difference between  the sum of the digits occuring in the alternate positions (starting from the first position) and the sum of the digits occuring in the alternate positions (startting from the last rightmost position).

**ANS:**

**JAVA:**

**import java.util.Scanner;**

**import java.lang.Math;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**int n = sc.nextInt();**

**int i = 1,sum\_even = 0,sum\_odd = 0;**

**while(n!=0){**

**if(i%2==0){**

**sum\_even+=(n%10);**

**n/=10;**

**i++;**

**}**

**else{**

**sum\_odd+=(n%10);**

**n/=10;**

**i++;**

**}**

**}**

**System.out.println(Math.abs(sum\_odd-sum\_even));**

**}**

**}**

**C:**

**#include <stdio.h>**

**#include <math.h>**

**int main(){**

**int n,sum\_e=0,sum\_o=0,i=1;**

**scanf("%d",&n);**

**while(n!=0){**

**if(i%2==0){**

**sum\_e+=(n%10);**

**n/=10;**

**i++;**

**}**

**else{**

**sum\_o+=(n%10);**

**n/=10;**

**i++;**

**}**

**}**

**printf("%d",abs(sum\_e-sum\_o));**

**}**

#### **Substrings and Sweets (D)**

Ramu, a small boy loves to eat sweets. Whenever he gets time, he goes to a local sweet shop and buys sweets, but his father is really worried about Ramu’s Health and wants to make sure that he doesn’t eat more sweets. So he indulged Ramu in an interesting game. Everyday he will give Ramu a word 'w' and a number 'k'. Ramu will be allowed to buy only the number of sweets equal to the number of substrings with 'k' vowels that he generates from the given word 'w' . Ramu wants to get the maximum number of possible sweets. But Ramu doesn’t know what a substring is and also do not know how to count numbers, fortunately, he knows that you are a good programmer, so he has given this task to you. Given the word 'w' and number 'k', write a code to print the number of sweets that Ramu can get.

**Vowels: a,e,i,o,u**

A **substring** is a contiguous sequence of characters within a string. For example, string apple all possible substrings of length greater than or equal to one are apple, appl, pple, app, ppl, ple, ap, pp, pl, le, a, p, p, l, e.

**Input Format:**

There are two lines of input,

The first line contains the word given by Ramu’s father and,

The second line contains the value of k.

**Output Format:**

Print only one integer, containing the value of sweets that Ramu can buy.

**Example:**

abc

1

All substrings of abc are:

 a, ab, abc, b, bc, c

Out of all these substrings a, ab and abc have 1 vowel exactly.

Hence the answer is 3.

**ANS.**

**JAVA:**

**import java.util.Scanner;**

**class Main{**

**static boolean vowel\_check(char c){**

**String vowels = "aeiouAEIOU";**

**return vowels.indexOf(c) != -1;**

**}**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**String s = sc.nextLine();**

**int l = s.length();**

**int k = sc.nextInt();**

**int ans = 0;**

**for(int i=0;i<l;i++){**

**for(int j=i;j<l+1;j++){**

**int count = 0;**

**String sub = s.substring(i,j);**

**for(int m=0;m<sub.length();m++){**

**if(vowel\_check(sub.charAt(m))){**

**count++;**

**}**

**}**

**if(count == k){**

**ans++;**

**}**

**}**

**}**

**System.out.println(ans);**

**}**

**}**

**C++:**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**bool vowel\_check(char c){**

**switch(c){**

**case 'a':**

**case 'e':**

**case 'i':**

**case 'o':**

**case 'u':**

**case 'A':**

**case 'E':**

**case 'I':**

**case 'O':**

**case 'U':**

**return true;**

**default: return false;**

**}**

**}**

**int main(){**

**int k,l,ans = 0;**

**string s;**

**cin>>s;**

**l = s.length();**

**cin>>k;**

**for(int i=0;i<l;i++){**

**for(int j=i;j<l+1;j++){**

**int count = 0;**

**string sub = s.substr(i,j-i);**

**for(int m=0;m<sub.length();m++){**

**if(vowel\_check(sub[m])){**

**count++;**

**}**

**}**

**if(count == k){**

**ans++;**

**}**

**}**

**}**

**cout<<ans;**

**}**

**PYTHON:**

**s = input()**

**k,ans = int(input()),0**

**for i in range(len(s)):**

**for j in range(i,len(s)+1):**

**c = 0**

**for m in s[i:j]:**

**if m.lower() in "aeiou": c+=1**

**if c==k: ans+=1**

**print(ans)**

#### **String Decoding**

Given an encoded string,you have to return it’s decoded string. The encoding rule is: k[encodedString], where the encoded\_string inside the square   
brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer greater than 1. You may assume that the input string is always valid; No extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there won’t be input like 3a or 2[4].

**Input Format**

First line of each input take a value of T, denoting number of testcases.  
First line of each testcase contains the encoded string.

**Output Format**

Each decoded string should be printed in new line.

**\*\*Sample Test case 0\*\***  
'''  
3  
3[a]2[bc]  
3[a2[c]]  
2[abc]3[cd]ef  
'''

**\*\*Output 0\*\***  
'''  
aaabcbc  
accaccacc  
abcabccdcdcdef  
'''

**ANS.**

**def decodeString(s):**

**stack=[]**

**for i in s:**

**if i!=']':**

**stack.append(i)**

**else:**

**temp=''**

**num=''**

**while stack[-1]!='[':**

**temp=stack.pop()+temp**

**stack.pop()**

**while len(stack)!=0 and stack[-1].isdigit():**

**num=stack.pop()+num**

**stack.append(int(num)\*temp)**

**res=''**

**for i in stack:**

**res+=i**

**return res**

**for i in range(int(input())): print(decodeString(input()))**

#### **GCD of (x, y, z)**

Greatest common divisor also referred as HCF (highest common factor).

For given three numbers x, y and z try to write a **RECURSIVE** program to find the GCD of those numbers.

HINT: - first find the GCD of x and y then then use that answer to find the its GCD with z.

**Input format:**\* First line contains N, number of testcases. Then testcases follows.

\* First and the only line of each testcase contains three space separated number mainly x, y and z.

**Output format:**\* Print GCD of each testcase in new line.

**ANS.**

**import java.util.\*;**

**class Main{**

**static int hcf(int n1, int n2){**

**if (n2 != 0)**

**return hcf(n2, n1 % n2);**

**else**

**return n1;**

**}**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**int t = sc.nextInt();**

**while(t-- > 0){**

**int x = sc.nextInt();**

**int y = sc.nextInt();**

**int z = sc.nextInt();**

**System.out.println(hcf(hcf(x,y),z));**

**}**

**}**

**}**

#### **Matrix sorting**

You are given a matrix of the form n X n, print the new matrix which you will be getting after performing sort operation in the following way, perform sorting of each row to get them arranged in ascending order and then sort each column to get them arranged in descending order.

**Input format:**

\* First line contains the value of n.

\* For the next n lines take n space separated values as input for the matrix.

**Output format**

\* Print the sorted matrix in n lines where each line contains

n space separated value.

**ANS.**

**#include <stdio.h>**

**void swap(int \*x,int \*y){**

**int t = \*x;**

**\*x = \*y;**

**\*y = t;**

**}**

**void sort(int arr[],int flag,int n){**

**if(flag == 0){**

**for(int i=0;i<n;i++)**

**for(int j=0;j<n-i-1;j++)**

**if(arr[j] > arr[j+1]) swap(&arr[j+1],&arr[j]);**

**}**

**if(flag == 1){**

**for(int i=0;i<n;i++)**

**for(int j=0;j<n-i-1;j++)**

**if(arr[j+1] > arr[j]) swap(&arr[j+1],&arr[j]);**

**}**

**}**

**int main(){**

**int n;**

**scanf("%d",&n);**

**int matrix[n][n];**

**for(int i=0;i<n;i++)**

**for(int j=0;j<n;j++)**

**scanf("%d",&matrix[i][j]);**

**for(int k=0;k<n;k++) sort(matrix[k],0,n);**

**int temp[n],m=0;**

**while(m < n){**

**for(int l=0;l<n;l++){**

**temp[l] = matrix[l][m];**

**}**

**sort(temp,1,n);**

**for(int l1=0;l1<n;l1++){**

**matrix[l1][m] = temp[l1];**

**}**

**m++;**

**}**

**for(int o=0;o<n;o++){**

**for(int p=0;p<n;p++){**

**if(p!=n-1) printf("%d ",matrix[o][p]);**

**else printf("%d",matrix[o][p]);**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

TEMP KO POINTER KA ARRAY BNA KE ALAG SE SORT FUNC BNA SKTE HAIN

#### **Registration Number**

Every student in VIT has his own unique registration number which follows a particular format that is NNWWWNNNN here N represents a number and W represents a word in capital letter. Take a registration number as an input and check whether its a valid registration number or not. Then even display the year in which the student with the given registration number will pass out if he/she is doing B.Tech. . For example: - 21BCE1846 Its a valid registration number and the student will pass out in 2025.

**Input format:**

* First and the only line contains the registration number as an input.

**Output format:**

* Print YES if the registration number is valid and also print the passing year in a new line else print NO.

**Constrain:**

* **2003 < passing out year < 3000**

**ANS.**

**PYTHON:**

**s,p\_year = input(),0**

**if s[:2].isnumeric():**

**p\_year = 2000+int(s[:2])+4**

**else:**

**print("NO")**

**exit()**

**for i in s[2:5]:**

**if i<'A' or i>'Z':**

**print("NO")**

**exit()**

**if s[5:].isnumeric() != True:**

**print("NO")**

**exit()**

**print("YES")**

**print(p\_year)**

**JAVA:**

**import java.util.\*;**

**import java.io.\*;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**try{**

**String s = sc.nextLine();**

**int year = Integer.parseInt(s.substring(0,2));**

**String batch = s.substring(2,5);**

**for(int i=0;i<batch.length();i++){**

**if(batch.charAt(i)>'Z' || batch.charAt(i)<'A'){**

**System.out.println("NO");**

**System.exit(0);**

**}**

**}**

**int roll\_no = Integer.parseInt(s.substring(5,s.length()));**

**System.out.println("YES");**

**System.out.println(year+2004);**

**}**

**catch(NumberFormatException e){**

**System.out.println("NO");**

**}**

**}**

**}**

**JAVA (REGEX):**

**import java.util.\*;**

**import java.util.regex.\*;**

**class Main**

**{**

**public static void main(String args[])**

**{**

**Scanner sc=new Scanner(System.in);**

**String rno=sc.next();**

**if(Pattern.matches("[0-9]{2}[A-Z]{3}[0-9]{4}",rno))**

**{**

**System.out.println("YES");**

**int year=Integer.parseInt("20"+rno.substring(0,2));**

**if(year>1998 && year<2995)**

**System.out.println(year+4);**

**}**

**else**

**{**

**System.out.println("NO");**

**}**

**}**

**}**

#### **Array Swapping**

For a given integer n take two arrays as an input with n space separated values. Iterate through both arrays and swap the elements such that all smaller elements go to array1 and larger elements goes to arrray2. For example: -

If n = 5 and

Array1 = [1,4,2,0,7]

Array2 = [2,3,5,1,6]

Then after swapping we will be getting: -

Array1 = [1,3,2,0,6]

Array2 = [2,4,5,1,7]

**Input format:**

* First line takes the value of n.
* Second line takes all the n elements of the array1 in space separated format.
* Third line takes all the n elements of the array2 in space separated format.

**Output format:**

* Print both arrays in two separate lines after swapping, all the elements of each array should be printed in a single line in space separated format.

**ANS.**

**#include <iostream>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**int n;**

**cin>>n;**

**int arr1[n],arr2[n];**

**for(int i=0;i<n;i++){**

**cin>>arr1[i];**

**}**

**for(int i=0;i<n;i++){**

**cin>>arr2[i];**

**}**

**for(int i=0;i<n;i++){**

**if(arr1[i]>arr2[i]){**

**swap(arr1[i],arr2[i]);**

**}**

**}**

**for(int i=0;i<n;i++){**

**if(i!=n-1) cout<<arr1[i]<<" ";**

**else cout<<arr1[i];**

**}**

**cout<<"\n";**

**for(int i=0;i<n;i++){**

**if(i!=n-1) cout<<arr2[i]<<" ";**

**else cout<<arr2[i];**

**}**

**}**

#### **Encryption**

Encryption is the process of converting normal message into meaningless message. Encrypt a given word using a given keyword while following the given method of encryption: -We know that the English alphabets are of the format A, B, C, D, E, F, G, H, …X, Y and Z, but here you will be provided with a keyword for instances “APPLE” so you have to consider the unique letter in the given keyword in the same order that is “APPLE” becomes “APLE”. Now this keyword will take the initial positions of the letters in English alphabets thus shifting others and finally we get A, P, L, E, B, C, D, F, G, H, I, J, K, M, N, O, Q, R, S, T, U, V, W, X, Y and Z. Using this we can convert the word “HAPPY” as “FAOOY”, because at the position of 'H' now we have 'F' and so on. So, the encrypted format of word “HAPPY” is “FAOOY” using the keyword “APPLE”.

**Input format:**

* First line takes the value of T denoting the number of test cases
* First line of each testcase two space separated strings as input the first one denotes the word to be encrypted and the other one denotes the keyword.

**Output format:**

* Print the encrypted word.

**ANS.**

**from collections import OrderedDict**

**for i in range(int(input())):**

**alp = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"**

**alp1 = list(alp)**

**word,key = map(str,input().split())**

**key = list(OrderedDict.fromkeys(key))**

**for x in key: alp1.remove(x)**

**alp2 = key+alp1**

**alp3 = list(alp)**

**info = dict(zip(alp3,alp2))**

**ans = ""**

**for i in word: ans+=info[i]**

**print(ans)**

#### **Dictionary Rank**

Write a computer program which prints the dictionary rank of a given word which is made up of all unique letters. For example:

Let us consider the word “MOTHER”. We start with the first lowest letter which is E. Thus, the letters made with E as the first letter would be 5! = 120.  
Similarly, the next lowest letter after E is H. Fix H at first place, again the number of combinations would be 5! = 120. After this, the next letter after H is M. Now, in this case, this aligns with the required word (MOTHER).  
Now, instead of calculating all the combinations. We first fix the second letter. Alphabetically, this would be E. Now, the total number of combinations with these two letters fixed are 4! = 24 then second letter alphabetically is H with same combination that is 4! = 24. The next available letter would be O. Again, MO aligns with the word ‘MOTHER’. So, now, we will try to fix 3 letters and then find the combinations. Again, alphabetically, we have E (thus, fixing MOE), we have total combinations as 3! = 6, same goes for H and R with combinations as 6 each. Now, the next letter would be T. This aligns with the word ‘MOTHER’. Now, we fix 4 letters. Thus, the first available letter would be E (MOTE is fixed). Thus, we have 2 combinations possible (2! = 2). Now, the next letter would be H, again MOTH aligns with the word MOTHER. Now, the next available letter would be E aligns with the word MOTHER. Thus, we have all the combinations.  
Sum up all combinations gives 120 + 120 + 24 + 24 + 6 + 6 + 6 + 2 = 308. Since, there are 308 combinations before the word, ‘MOTHER’. The rank of the word MOTHER is 308 + 1 =309.

**Input format:**

* First line takes the word as input.

**Output format:**

* Print the dictionary rank of the given word

**ANS.**

**from math import factorial**

**s = list(input())**

**x = sorted(s)**

**fact\_num = len(s)-1**

**priority,step = [],[]**

**for i in s:**

**priority.append(x.index(i)+1)**

**for i in range(len(priority)):**

**count = 0**

**for j in range(i+1,len(priority)):**

**if priority[j] < priority[i]:**

**count+=1**

**step.append(count)**

**rank = 0**

**while(fact\_num >= 0):**

**rank+=factorial(fact\_num)\*step.pop(0)**

**fact\_num-=1**

**rank+=1**

**print(rank)**

#### **Selection of lucky winner(D)**

In a cinema theatre named 'w', with `n' seats, whenever the theatre is full (i.e., when all the seats are occupied), theatre administartion will be gifting a viewer with a surprise gift and the lucky viewer will be chosen through the following process, called as the  'Character allottment process'.

**Character allotment process :**

1. Starting from the first seat, allot a character  of `w' ( starting from the first character) to each seat.  That is, allot the first character of w to the seat no.1, allot the second character of  `w' to the seat no.2 and so on.

2. If all the characters of `w' are allotted, then continue the allotment from the first symbol of `w' again.

3. After allotting a character to the last seat, continue the allotment process by allotting the next character (character,  next to the character allotted to the last seat earlier) from the last seat

4. Continue allotting the characters  till the first seat.

5. In the process, all the seats would have two characters allotted to each seat, one character during the forward allottment process  (when the characters are allotted from the first seat to the last seat) and another during the backward allottement process. (when the characters are allotted from the last seat to the first seat)

6. The first seat which gets the same character during both the forward allottment process and the backward allottment process is the lucky seat.

7. The person occupying the luck seat is given the surprise gift.

For eg, Name of the theatre : good.  Total number of seats : 10  During the forward allottment, seats are allotted as : 1-g, 2-o, 3-o, 4-d,5-g, 6-o, 7-o, 8-d, 9-g, 10-o

and during reverse allotment process 10-o, 9-d, 8-g, 7-o, 6-o, 5-d, 4-g, 3-o, 2-o, 1-d. The first seat to get the same characters in both the allottment is the seat no : 10.  The person who occupied the seat no. 10 is the lucky winner.

Given the name of the theatre `w', the total number of seats in the theatre, write an algorithm and the code to identify the lucky seat. Print -1 if no seat number could get the same letters.

**Input format :**

Name of the theatre: w

Total number of  seats in the theatre: n

**Output format :**

The seat number of the lucky seat.

**ANS.**

**w,n = input(),int(input())**

**info1,info2 = {},{}**

**nums1 = [i for i in range(1,n+1)]**

**nums2 = [i for i in range(n,0,-1)]**

**str1,str2 = "",""**

**for i in range(n//len(w)): str1+=w**

**if(n%len(w)!=0):**

**str1+=w[:n%len(w)]**

**str2+=w[n%len(w):]**

**for i in range((n-(n%len(w)))//len(w)): str2+=w**

**else: str2 = str1**

**str1,str2 = list(str1),list(str2)**

**info1 = dict(zip(nums1,str1))**

**info2 = dict(zip(nums2,str2))**

**for i in info2.keys():**

**if info2[i] == info1[i]: print(i); exit()**

**print(-1)**

**shorter code:**

**w,n = input(),int(input())**

**ans1,ans2 = "",""**

**for i in range(n//len(w)): ans1+=w**

**if n%len(w) != 0:**

**ans1+=w[:n%len(w)]**

**ans2+=w[n%len(w):]**

**for i in range(n-(n//len(w))\*len(w)): ans2+=w**

**else: ans2 = ans1**

**key = [i for i in range(1,n+1)]**

**d1,d2 = dict(zip(key,list(ans1))),dict(zip(key[::-1],list(ans2)))**

**for i in d2:**

**if d1[i] == d2[i]: print(i); exit()**

**print(-1)**

#### **Dominant Number (D)**

Given two numbers print the dominant number. A number is said to be dominant over the other if the sum of proper factors of the number is greater than the sum of proper factors of the other. Factors of a number excluding 1 and itself are called proper factors. Print 'No dominance' when the sum of the proper factors are same.

For example, fourteen is dominant over fifteen as sum of factors of 14 is 2+7 = 9 whereas sum of factors of 15 is 3+5 = 8. Whereas when the numbers 6 and 25 are considered there is 'No dominance' as they have the same sum of proper factors for the number 6 it is 2+3 = 5 and for the number 25 also it is 5.

**Input Format**

First line contains the first number, m

Second line contains the second number, n

**Output Format**

When sum of factors of numbers m and n are no equal, print the number with larger sum of proper factors and print No dominance when the sum of proper factors of m and n are equal

**ANS.**

**m,n = int(input()),int(input())**

**sum1,sum2 = 0,0**

**for i in range(2,m//2+1):**

**if m%i == 0: sum1+=i**

**for i in range(2,n//2+1):**

**if n%i == 0: sum2+=i**

**if(sum1>sum2): print(m)**

**elif(sum1<sum2): print(n)**

**else: print("No dominance")**

**better solution:**

**from math import sqrt**

**m,n = int(input()),int(input())**

**sum1,sum2 = 0,0**

**for i in range(2,int(sqrt(max(m,n)))+1):**

**if m%i == 0 and i <= int(sqrt(m)):**

**if i != m//i:**

**sum1+=i; sum1+=m//i**

**else: sum1+=i**

**if n%i == 0 and i <= int(sqrt(n)):**

**if i != n//i:**

**sum2+=i; sum2+=n//i**

**else: sum2+=i**

**if sum1 > sum2: print(m)**

**elif sum1 < sum2: print(n)**

**else: print("No dominance")**

#### **Consecutive Four Sum Number**

A positive integer is called as a `Consecutive Four Sum (CFS) number' if that number can be expressed as the sum of four consecutive positive integers.  10 is a CFS number since 10= 1+2+3+4; 50 is a CFS number since 50= 11+12+13+14.  7 is not a CFS number since  7 cannot be expressed as a sum of four consecutive number.  Similarly, 20 is not a CFS number.  Given a number `n', write an algorithm and the subsequent python code to check whether the given number is a CFS number or not. If so, your code should  print 1 ( here, 1 is an indicator which conveys that the given number is a CFS number)  and also print the starting integer among the four consecutive integers.  If the given number is not a CFS number, your code should output 0.  For eg, if 10 is the input, your code should output : 1 and 1.  Here 1 is the starting integer of the four consecutive integers for 10. If 7 is the input, your code should output : 0.  Given the integer 20, your code should output only 0.

**Input format:**

Enter the positive integer

**Output format :**

0 or 1   (If the given number is a CFS , output should be 1;  If the given number is not a CFS, it should be 0.)

Starting integer of the four consecutive integers. ( The second output, is applicable only when the first output is 1)

**Sample Input :**

10

**Sample Output**

1

1

**Note :** The two outputs should be printed in two separate lines

**ANS.**

**n = int(input())**

**a = ((n/2)-3)/2**

**if(int(a) == a): print(1,int(a),sep = "\n")**

**else: print(0)**

**C++**

**#include <iostream>**

**using namespace std;**

**int main(){**

**float n;**

**cin>>n;**

**float a = ((n/2)-3)/2;**

**if(a == (int)a) cout<<1<<"\n"<<int(a);**

**else cout<<0;**

**}**

**JAVA:**

**import java.util.\*;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**double n = sc.nextDouble();**

**double a = ((n/2)-3)/2;**

**if(a == (int)a){**

**System.out.println(1);**

**System.out.println((int)a);**

**}**

**else System.out.println(0);**

**}**

**}**

#### **Unique Dimension Rectangles (UDRs) (D)**

A collection of rectangles are said to be unique dimension rectangles (UDRs) if they have distict dimensions. Dimension of a rectangle is a pair of length and breadth represented as (length, breadth). That is a collection of five rectangles with integer dimensions (10, 20), (34, 56), (12, 13), (19,10), (16, 18) are UDRs and whereas the five rectangles with the dimension (10, 20), (34, 56), (12, 13), (19,10), (20, 10) cannot be UDRs since (10, 20) and (20,10) are duplicated except change in order. Given an integer 'n' write a code to findout the number of UDRs that can be formed with area as 'n' and also print the dimensions.

For example if 168 is given then there are eight UDRs with dimensions as follows (1,168), (2, 84), (3, 56), (4,42), (6, 28), (7, 24), (8,21), (12,14).

**Input Format**

First line contains an integer, n

**Output Format**

First line contains the number of UDRs that can be formed, u

Next ‘u’ lines contains the pair of numbers in ascending order separated by a space and that can be dimension of rectangles whose area is n

**ANS.**

**n,udrs = int(input()),[]**

**for i in range(1,n//2):**

**if n%i == 0 and i<n//i : udrs.append((i,n//i))**

**print(len(udrs))**

**for i in udrs: print(\*i)**

**JAVA:**

**import java.util.\*;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**int n = sc.nextInt();**

**ArrayList <int []> udrs = new ArrayList <int []> ();**

**for(int i=1;i<=n/2;i++){**

**int pair[] = new int[2];**

**if(n%i==0 && i<n/i){**

**pair[0] = i;**

**pair[1] = n/i;**

**udrs.add(pair);**

**}**

**}**

**System.out.println(udrs.size());**

**for(int[] p1: udrs){**

**System.out.print(p1[0]+" "+p1[1]+"\n");**

**}**

**}**

**}**

**More efficient approach in cpp:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**#include <cmath>**

**#define FOR(i,a,n) for(int i=a;i<n;i++)**

**#define vp vector <pair<int,int>>**

**#define pb push\_back**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**vp list; int cnt = 0;**

**FOR(i,1,(int)sqrt(n)+1){**

**if(n%i == 0 && n/i != i){**

**list.pb({i,n/i}); cnt++;**

**}**

**}**

**sort(list.begin(),list.end());**

**cout<<cnt<<endl;**

**for(auto x: list){**

**cout<<x.first<<" "<<x.second<<endl;**

**}**

**}**

#### **Points on the line (D)**

A line in 2-dimensional plane is represented as an equation a\*x+b\*y+c=0, where `a’ is called the coefficient of x, `b’ is called as the coefficient of y and `c’ is the constant term. Here a,b, c are all real numbers. A point in a 2-dimensional plane is represented as a pair of numbers (x1, y1), where x1 and y1 are both real numbers and x1 is called as the x-coordinate of the point , y1 is called as the y-coordinate of the point.

A point (x1,y1) will lie  on a line if a\*(x1) +b ( y1) +c =0. Consider the line 2\*x+3\*y-1=0. The point (-1,1) is a point on the line : 2\*x+3\*y-1=0 since 2\* (-1) +3\*(1)-1=0.

Given an equation of the line a\*x+b\*y+c=0 and an integer n, write an algorithm and the subsequent code to print n points on the line such that the x-coordinates of all the n-points are the odd integers 1,3,5,7,…. respectively. Among the n-points , x-coordinate of the first point is 1, x-coordinate of the second point is 3, x-coordinate of the third point is 5 and so on. Your code should print the y-coordinates of all the n-points that lie on the given line. All the real numbers are represented in the 2-decimal format.

Let the equation of the line 2\*x + 3 \* y -15 =0, 3 points which lie on this line, are required such that the x-coordinate of the first point is 1, the x-coordinate of the second point is 3, x-coordinate of the third point is 5.

Here 2\*1+3\*4.33-15=0. Hence, the point (1,4.33) lie on the line

2\*3 +3\*(3.00)-15=0. Hence, the point (3,3.00) lie on the line

2\*5 +3\*(1.67)-15=0. Hence the point (5, 1.67) lie on the line.

Hence, your program should output 4.33, 3.00. 1.67

**Note:**To print the only decimal places of value of a variable answer, syntax to be used in Python is

print(format(answer,'0.2f'))

**Input format :**

First line contains the coeeficient of x, a

Second line contains the coefficient of y, b

Third line contains the value of constant term , c

Fourth line contains the number of points required , n

**Output format:**

First line contains the y-coordinate of the first point

Second line contains y-coordinate of the second point

…**..**

…**..**

nth line contains y-coordinate of the n-th point

**Example :**

**Input :**

2

3

-15

3

**Output :**

4.33

3.00

1.67

**ANS.**

**a,b,c,n,i,count = int(input()),int(input()),int(input()),int(input()),1,0**

**while(count != n):**

**print(format((-c-(a\*i))/b,'0.2f'))**

**i+=2; count+=1**

**JAVA:**

**import java.util.\*;**

**import java.lang.\*;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**double a = sc.nextDouble();**

**double b = sc.nextDouble(); //THESE THREE MUST BE DOUBLE FOR FORMAT TO WORK PROPERLY**

**double c = sc.nextDouble();**

**int n = sc.nextInt();**

**int i = 1,count = 0;**

**while(count < n){**

**double d = (-c-a\*i)/b;**

**System.out.println(String.format("%.2f",d));**

**i+=2;**

**count++;**

**}**

**}**

**}**

**Better approach (cpp):**

**#include <iostream>**

**using namespace std;**

**int main(){**

**float a,b,c,n,i=1;**

**cin>>a>>b>>c>>n;**

**while(i/2 < n){**

**printf("%.2f\n",(a\*i+c)/(-1\*b)); i+=2;**

**}**

**}**

#### **Reverse Length Divisible (D)**

Given a number **N,** check if it is reverse length divisible. A number is said to be reverse length divisible if the first i digits of the number is divisible by (l-i-1) where l is the number of digits in **N** and **0 < i <= l.**

For example, 52267 is reverse length divisible because:

5 is divisible by 5

52 is divisible by 4

522 is divisible by 3

5226 is divisible by 2

52267 is divisible by 1.

43268 is not reverse length divisible. Print Yes if the number is reverse length divisible and no otherwise.

**Note:**

* It is easy to solve this problem by using strings, but if you want some challenge, solve this problem without strings or any library functions.
* Be cautious about choosing the data types and the boundary conditions

**Boundary Conditions**

0<n<10000000

**Input Format**  
The first line contains the number **N**

**Output Format**  
Print "Yes" if the number is reverse length divisible otherwise print "No" (without quotes)

**ANS.**

**#include <iostream>**

**#include <cmath>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**int n;**

**cin>>n;**

**vector <int> digits;**

**while(n!=0){**

**digits.push\_back(n%10);**

**n/=10;**

**}**

**reverse(digits.begin(),digits.end());**

**for(int i = 1;i<digits.size()+1;i++){**

**int num = 0;**

**for(int j = 0;j<i;j++){**

**num+=digits[j]\*pow(10,i-j-1);**

**}**

**if(num%(digits.size()-i+1) != 0){**

**cout<<"No";**

**exit(0);**

**}**

**}**

**cout<<"Yes";**

**}**

#### **Rumbling**

Eren Yeager wants to activate Rumbling, for which he must find number of pairs (A, B) using given integer N that follows the given condition.  
• 1≤A, B≤N  
• A+B is odd

**Boundary Conditions**  
N<10000

**Input Format**  
The first line contains the value of N

**Output Format**  
Number of pairs which follow the condition

**ANS.**

**n,count = int(input()),0**

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

**for j in range(1,n+1):**

**if (i+j)%2 != 0: count+=1**

**print(count)**

**C++**

**#include <iostream>**

**using namespace std;**

**int main(){**

**int n,count = 0;**

**cin>>n;**

**for(int i=1;i<=n;i++)**

**for(int j=1;j<=n;j++)**

**if((i+j)%2) count++;**

**cout<<count;**

**}**

#### **Right Digit Numbers**

* Given a four digit number, check if it is a right digit number.
* A number is right

digit if the number formed by 10s and 1s place is divisible by the number formed by 1000s

and 100s place.

* If it is a right digit number print, YES else print NO.

**Input Format**

* The first line contains the 4-digit number

**Output Format**

YES, if valid, else NO

**ANS.**

**import java.util.\*;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**String n = sc.next();**

**int n1 = (int)(n.charAt(2)-'0')+10\*(int)(n.charAt(3)-'0');**

**int n2 = (int)(n.charAt(1)-'0')+10\*(int)(n.charAt(0)-'0');**

**System.out.println(n1%n2==0?"YES":"NO");**

**}**

**}**

**PYTHON:**

**n = input()**

**if(int(n[2:])%int(n[:2])): print("NO")**

**else: print("YES")**

**C:**

**#include <stdio.h>**

**int main(){**

**int n;**

**scanf("%d",&n);**

**int arr[4],i = -1;**

**while(n != 0){**

**arr[++i] = n%10;**

**n/=10;**

**}**

**int n1 = arr[1]\*10+arr[0];**

**int n2 = arr[3]\*10+arr[2];**

**printf(n1%n2?"NO":"YES");**

**}**

#### **Levi vs Titans**

* Levi has to kill the beast titan for which he has to reach its Nape.If he is short of distance the Beast titan will kill him.If he goes past his nape,the Jaw titan will kill him.
* Given his starting position,number of moves made by him,distance moved by him in each move and position of Beast titan, find if he kills Beast titan or not.
* If Levi kills the Beast ,print Levi.
* If he falls short of the distance, print Beast
* If he goes past the Beast Titan, print Jaw

**Input Format**

The first line contains position of Levi

Second line contains number of moves made by Levi

Third line contains the distance covered by Levi in each move separated by a space

Fourth line contains position of the Beast titan.

**Output Format**

Levi / Beast /Jaw

**ANS.**

**PYTHON:**

**levi = int(input())**

**no\_of\_moves = int(input())**

**moves = tuple(map(int,input().split()))**

**beast = int(input())**

**levi+=sum(moves)**

**if(levi < beast): print("Beast")**

**elif(levi > beast): print("Jaw")**

**else: print("Levi")**

**C++:**

**#include <iostream>**

**using namespace std;**

**int main(){**

**int levi,no\_of\_moves,sum = 0,m,beast;**

**cin>>levi;**

**cin>>no\_of\_moves;**

**for(int i=0;i<no\_of\_moves;i++){**

**cin>>m;**

**sum+=m;**

**}**

**cin>>beast;**

**levi+=sum;**

**if(beast == levi) cout<<"Levi";**

**else if(beast > levi) cout<<"Beast";**

**else cout<<"Jaw";**

**}**

#### **Distinct Array**

Let F(S) denote the number of **distinct** elements in the array S. For example, F([1, 2, 3, 2]) = 3, F([1, 2]) = 2.

You are given an array A containing N integers. Find if it is possible to divide the elements of the array A into two arrays B and C such that :

* Every element of the array A belongs either to array B or to array C.
* F(B) = F(C).

### **Input Format**

* The first line of input will contain a single integer T, denoting the number of test cases.
* Each test case consists of two lines of input.
  + The first line of each test case contains an integer N — the length of the array A.
  + The next line contains N space-separated integer A1​, A2​,…,AN​, denoting the elements of the array A.

### **Output Format**

For each test case, print YES if it is possible to divide the elements of the array A into two arrays B, C satisfying all the conditions and NO otherwise.

### **Constraints**

* 1 ≤ T ≤ 104
* 2 ≤ N ≤ 105
* 1 ≤ Ai ​≤ N
* The sum of N over all test cases won't exceed 2⋅105.

**ANS.**

**t = int(input())**

**for i in range(t):**

**n = int(input())**

**arr = list(map(int,input().split()))**

**arr1 = sorted(arr)**

**flag = 0**

**for i in range(1,len(arr)):**

**if(len(set(arr[:i])) == len(set(arr[i:])) or len(set(arr1[i:]))==len(set(arr1[:i]))):**

**print("YES")**

**flag = 1**

**break**

**if(flag == 0): print("NO")**

#### **Manipulating the list**

You are given a sorted list X of size n. You have to make a new list Y such that Y[i] is equal to the number of elements strictly greater than X[i] in list X.

Print the new list.

**Input Format**

The first line of each test case consists of one integer denoting n, where n is the size of the list given to you.

The second line of each test case contains the list given to you containing n elements.

**Output format**

In a single line print all the elements of the array Y in a space separated format.

**ANS.**

**n,l = int(input()),list(map(int,input().split()))**

**for i in range(len(l)):**

**count = 0**

**for j in range(i+1,len(l)):**

**if(l[i]<l[j]): count+=1**

**print(count,end = " ")**

**JAVA:**

**import java.util.\*;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**int n = sc.nextInt();**

**int[] l = new int[n];**

**for(int i=0;i<n;i++) l[i] = sc.nextInt();**

**for(int i=0;i<n;i++){**

**int count = 0;**

**for(int j=i+1;j<n;j++){**

**if(l[i]<l[j]) count++;**

**}**

**System.out.print(count+" ");**

**}**

**}**

**}**

#### **Candy Game**

Bears love candies and games involving eating them. Limak and Bob play the following game. Limak eats 1 candy, then Bob eats 2 candies, then Limak eats 3 candies, then Bob eats 4 candies, and so on. Once someone can't eat what he is supposed to eat, he loses.

Limak can eat at most A candies in total (otherwise he would become sick), while Bob can eat at most B candies in total. Who will win the game? Print "Limak" or "Bob" accordingly.

**Input Format**

Each test case contains two integers **A**and **B**denoting the maximum possible number of candies Limak can eat and the maximum possible number of candies Bob can eat respectively.

**Output Format**

Name of the winner

**ANS.**

**a,b = map(int,input().split())**

**chance\_present,chance\_next,i = 1,2,1**

**while(True):**

**if i%2 == 1:**

**if a<0: break**

**a-=i**

**i+=1**

**chance\_present,chance\_next = chance\_next,chance\_present**

**else:**

**if b<0: break**

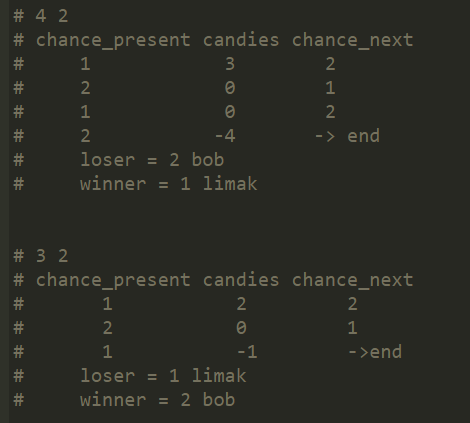
**b-=i**

**i+=1**

**chance\_present,chance\_next = chance\_next,chance\_present**

**if chance\_present == 1: print("Bob")**

**else: print("Limak")**

****

**O(1) solution:**

**a,b = map(int,input().split())**

**x = int(a\*\*0.5)\*\*2**

**if x<b: print("Bob")**

**else: print("Limak")**

**My O(1) solution:**

**a,b = map(int,input().split())**

**if a<b: print("Bob") #if b is greater then bob wins**

**elif a>b and int(a\*\*0.5)\*\*2 < b: print("Bob") # if sum of first n odd numbers is less than b then bob wins**

**else: print("Limak") # in rest cases limak wins**

#### **Eren Yeager and Survey Corps**

Eren Yeager after seeing his mother being devoured by a Titan, he decides to dedicate his life to eradication of Titan by joining Survey Corps. For this he must go for 104th Training Corps where along with many others will be trained and educated to become the member of three military branches namely Garrison Regiment, Survey Corps and Military Police. For this Eren must get into top three candidates among other students. For a given number of students and their performance in physical, physiological and group activities find whether Eren will be able to make up to top three positions or not. Print YES or NO accordingly.

### **Input Format**

* The first line will take n i.e., number of students in one batch will be taken as input.
* For next three line take n space separated integers values denoting the performance of students in physical, physiological and group activities respectively out of 10. (Every First value is the marks scored by Eren Yeager in the respective exam)

### **Output Format**

### **Find the sum of all the marks and check whether Eren will be able to make up to top three candidates or not. Print YES or No accordingly.**

**ANS.**

**n = int(input())**

**phy = list(map(int,input().split()))**

**physio = list(map(int,input().split()))**

**grp = list(map(int,input().split()))**

**perf = []**

**for i in range(n):**

**perf.append(phy[i]+physio[i]+grp[i])**

**perf\_sorted = sorted(perf)**

**if perf\_sorted[-1] == perf[0] or perf\_sorted[-2] == perf[0] or perf\_sorted[-3] == perf[0]:**

**print("YES")**

**else: print("NO")**

**Eren Yeager Vs Armor Titan**

The day has finally come Eren has to face Armor Titan and defeat him or else he would be killed.

Eren must fight in order to defend his own life.

Check if Eren punch Armor Titan at a specific spot with a specific power will he be able to take

Armor Titan down.

**Input Format**

First line of input takes n (number of weak spots) as an input.

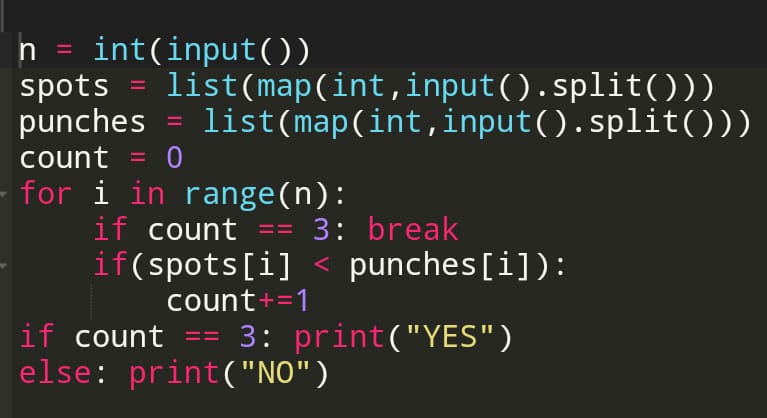
Second line takes n space separated integers as input denoting the strength of each spot in Armor Titan body.

Third line takes n space separated integers as input denoting the strength of each punch of Eren.

**Output Format**

Check if Eren could break at least three weak spots of Armor Titan or not (strength of a spot of Armor Titan should be less than Eren’s Punch). Print YES or NO accordingly.

**ANS.**

****

**To Divide Or Not**

Alice likes all the numbers which are divisible by A. Bob does not like the numbers which are divisible by B and likes all the remaining numbers.

Determine the smallest number greater than or equal to N which is liked by both Alice and Bob.

Output -1 if no such number exists.

**Input Format**

The first and only line of each test case contains three space-separated integers A, B, and N — the parameters mentioned in the problem statement.

**Output Format**

Output the smallest number >= N which is divisible by A and is not divisible by B.

Output -1 if no such number exists.

**ANS.**

**a,b,n = map(int,input().split())**

**flag = 0**

**if(a==b or b==1): print(-1); exit()**

**if(n%a != 0): n = n-(n%a)+a**

**while(flag != 1):**

**if(n%b != 0):**

**flag = 1**

**print(n)**

**else:**

**n+=a**

#### **Hungry Naruto**

Naruto is very hungry. So, Naruto goes to a shop selling burgers. The shop has 2 types of burgers:

Normal burgers, which cost X rupees each

Premium burgers, which cost Y rupees each (where Y > X)

Naruto has R rupees. Naruto wants to buy exactly N burgers. He also wants to maximize the number of premium burgers he buys. Determine the number of burgers of both types Naruto must buy.

Output -1 if it is not possible for Naruto to buy N burgers.

**Input Format**

The first and only line of each test case contains four space-separated integers X, Y, N, and R — the cost of a normal burger, the cost of a premium burger, the number of burgers Naruto wants to buy and the amount of money Naruto has.

​**Output Format**

For each test case, output on a new line two integers: the number of normal burgers and the number of premium burgers Naruto must buy satisfying the given conditions.

Output -1 if he cannot buy N burgers.

**ANS.**

**x,y,n,r = map(int,input().split())**

**max\_r,max\_premium,count = 0,0,0**

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

**cost = x\*(n-i)+y\*i**

**if(r >= cost):**

**count += 1**

**if max\_r < cost:**

**max\_r = cost**

**max\_premium = i**

**elif max\_r == cost:**

**if max\_premium < i:**

**max\_premium = i**

**if count == 0: print(-1)**

**else: print(n-max\_premium,max\_premium)**

#### **IP Address**

Suppose you are an intern at a Cyber Security Lab. You are given some IP Addresses and you need to check if it is in the correct format to be used as a proxy or not.  
  
For an IP Address to be correct it should satisfy the following conditions:

* It should contain 4 integers, separated by a full stop (.)
* It cannot have spaces.
* All the integers must be in the range 0 to 255

Hint: Take IP as String Input then split them.

**Input Format:**

First line contains the number of test cases.  
Each contain a single line input of an IP Address

**Output Format:**

If the IP is correct Print “**Can Be Used as a Proxy**”.

If not then print “**Cannot be used as a Proxy**”.

**ANS.**

**JAVA:**

**import java.util.\*;**

**class Main**

**{**

**public static void main(String args[])**

**{**

**Scanner sc = new Scanner(System.in);**

**int n = sc.nextInt();**

**for(int i=0;i<n;i++)**

**{**

**String ip = sc.next();**

**String[] arr = ip.split("[.]");**

**int flag = 0;**

**try{**

**for(String a: arr)**

**{**

**if(Integer.valueOf(a) > 255 || Integer.valueOf(a) < 0)**

**{**

**flag = 1;**

**break;**

**}**

**}**

**}**

**catch(NumberFormatException e)**

**{**

**flag = 1;**

**}**

**if(flag == 1) System.out.println("Cannot Be Used as a Proxy");**

**else System.out.println("Can Be Used as a Proxy");**

**}**

**}**

**}**

**PYTHON:**

**for \_ in range(int(input())):**

**ip = input().split('.')**

**f = 0**

**try:**

**for i in ip:**

**x = int(i)**

**if x>255 or x<0: print("Cannot Be Used as a Proxy"); f = 1; break**

**except: print("Cannot Be Used as a Proxy"); f = 1**

**if f == 0 : print("Can Be Used as a Proxy")**

#### **Levi Vs Beast Titan**

Levi being surrounded by many other medium size titans and he is aiming to kill the Beast Titan. For him to do this he has to kill all other Titan within a limited amount of gas. Given the number of titans, amount of gas he has and the number of time Levi jumps by using gas machine to kill each titan find whether at the end he would be left with enough gas to kill the Beast Titan or not. Note with every jump 0.5 litre of gas is used.

### **Input Format**

* First line contains two space separated values n and l namely number of titans and amount of gas Levi has initially.
* Second line contains n space separated values denoting number of times Levi must jump to kill each titan.

**Output Format**  
•    Print YES or No based on the fact that Levi will be able to defeat the Beast Titan or not.

**ANS.**

**n,l = map(int,input().split())**

**jumps = list(map(int,input().split()))**

**if(sum(jumps)\*0.5 < l): print("YES")**

**else: print("NO")**

**C++**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main(){**

**int n,l,sum = 0;**

**cin>>n>>l;**

**vector <int> jumps(n);**

**for(auto x: jumps){**

**cin>>x;**

**sum+=x;**

**}**

**if(sum\*0.5 < l) cout<<"YES";**

**else cout<<"NO";**

**}**

#### **Secondary Factorial (D)**

Factorial of an integer n,  denoted as n! is defined as the product of the first n natural numbers

n! = 1\*2\*3....\*n

1! = 1 and 0! = 1

We define a secondary factorial of a number n, denoted by SF(n), as follows:

SF(n) = 1\*3\*5\*....\*n, if n is odd and

SF(n) = 2\*4\*6\*....\*n if n is even

If n is an odd number, SF(n) is defined as the product of all the odd numbers, starting from 1, till the number n. SF(5)= 1\*3\*5= 15.

If n is an even number, SF(n) is defined as the product of all the even numbers, starting from 2, till the number n. SF(6)=2\*4\*6=48.

Given a number k, write a code to compute SF(n), where k = n!.  For the given number k, If there is no number n such that  n! = k then, your code should print -1.

**Illstration**

Given k = 24 then 24 = 4! and

SF(4) = 2\* 4 = 8.

Given k=25, there is no number n such that 25 = n!, then the out put should be -1.

Given k=6, 6=3!. SF(3)=1\*3=3

**Input Format**

First line contains an integer, k

**Ouput Format**

Print SF(n) if there exists a number n, such that k = n! and -1 otherwise

**ANS.**

**def SF(n):**

**if n%2:**

**if n==1: return 1**

**else: return n\*SF(n-2)**

**else:**

**if n==2: return 2**

**else: return n\*SF(n-2)**

**def fact(n):**

**if n==1: return 1**

**else: return n\*fact(n-1)**

**n = int(input())**

**if n == 1: print(SF(1))**

**elif n == 2: print(SF(2))**

**elif n == 6: print(SF(3))**

**elif n == 24: print(SF(4))**

**elif n == 120: print(SF(5))**

**elif n == 720: print(SF(6))**

**else:**

**for i in range(7,n//2+1):**

**if fact(i) == n: print(SF(i)); exit()**

**print(-1)**

#### **Togetherness of a Family**

In the world of magic land, there is a park in which a group of people are playing a game. Every person in the group is a child of one another person in the group except one person who is the parent of all the people in the group. Here, each person is represented by a number that if there are ‘n’ persons in the group then they are represented by numbers 1 to 7. The direct and indirect child relationship is defined as follows:

(i) A person ‘b’ is said to be direct child of ‘a’ if ‘a’ is parent of ‘b’.

(ii) Person is ‘c’ is said to be indirect child of ‘a’ when ‘c’ is child of ‘b’ and ‘b’ is child of ‘a’.

(iii) A indirect child ‘i’ of a person ‘p’ is also an indirect child of parent of ‘p’.

In the game, everyone is given a token which has a number written on it and to test the togetherness of the family, the family is given some queries.

In one query, a number representing a person is given and it is asked to find the number of direct and indirect children of the person who have prime number in the token given to them.

For more clarifications see the example below.

There are seven person in the group and the tokens assigned to each of them are:

|  |  |
| --- | --- |
| **Person** | **Token Number** |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |

We are given 2 queries:

Query1: The person numbered as 1 is given and we need to findout the number of children(both direct and indirect under 1) which have a token with prime number value:

Answer: 4.

Explanation:

For the person numbered as 1, its direct children are persons numbered as 2 and 3 and they have tokens 2 and 3 respectively which are prime numbers.

The person numbered as 5 and 7 are also children (indirect) of the person number 1 hence they also added up to the count and hence there are a total of 4 people who are children of person number 1 and have a prime number token.

Query2: The person numbered as 6 is given

Answer: 0.

Explanation:

The person with number 6 has no children and hence there are 0 children under him having tokens with prime number values.

**Input Format:**

First line of the input contains N, the number of persons playing in the park

Second line contains token numbers assigned for the persons and they are separated by a space ith number in the line contains the value of the token assigned to the ith person.

Next N-1 lines contains two integers Xi and Yi separated by a space and indicates that Xi is the parent of Yi.

The next line contains the number of queries, Q

Next Q lines contain the person queried

**Note:** The persons are indexed from 1.

**Output Format:**

Output Q lines, each containing only 1 integer, the number of children for the given node, which have their token value as a prime number

Example:

Input:

7

1 2 3 4 5 6 7

1 2

1 3

2 4

2 5

3 6

3 7

2

1

6

Output:

4

0

As can be seen

1 is the parent of both 2 and 3.

2 is the parent of 4 and 5.

3 is the parent of 6 and 7

**ANS.**

**from math import sqrt**

**def prime(n):**

**if n == 1: return 0**

**for i in range(2,int(sqrt(n))+1):**

**if n%i == 0: return 0**

**return 1**

**n = int(input())**

**tokens = list(map(int,input().split()))**

**tree = {}**

**for i in range(n+1): tree[i+1] = []**

**result = [0]\*(n+1)**

**for i in range(n-1):**

**x,y = map(int,input().split())**

**if x in tree: tree[x].append(y)**

**else: tree[x] = y**

**def dfs(source):**

**for child in tree[source]: dfs(child)**

**for child in tree[source]:**

**if prime(tokens[child-1]): result[source]+=1**

**result[source] += result[child]**

**dfs(1)**

**q = int(input())**

**for i in range(q):**

**query = int(input())**

**print(result[query])**

#### **Maxpower (D)**

A new operation called maxpower, among two real numbers x , y, denoted by \*\*\*, is defined as follows.

x\*\*\*y = z, where y^z =x.

x\*\*\*y is the power to which a number y must be raised in order to get x.

100\*\*\*10 = 2 , since 10 ^ 2 = 100.

2\*\*\*10 = 0.3010, since 10 ^0.3010 = 1.99 = 2 (approximately), when rounded-off to the nearest integer,

10\*\*\*2= 3.3219 since 2^3.3219 = 9.99=10 (approximately), when rounded-off to the nearest integer.

64\*\*\*4 =3, since 4^3=64.

1.44 \*\*\*1.2 = 2, since 1.2^2= 1.44

Given two real numbers x and y, write a code to print the value of x\*\*\*y with two decimal places only.

**Boundary Conditions**

x, y >= 0

**Input format:**

The first line is the number x.

The second line is the number y

**Output format**

Value of the x\*\*\*y with two decimal places only

**ANS.**

**#include <iostream>**

**#include <cmath>**

**#include <iomanip>**

**using namespace std;**

**int main(){**

**int x,y;**

**cin>>x>>y;**

**cout<<fixed<<setprecision(2)<<log(x)/log(y);**

**}**

#### **Maximum Sized Lines**

A horizontal line is a line which connects two points whose y-coordinates are same. For example, the line connecting the points (1,2) and (5,2) is a horizontal line of size 1.

Similarly, A vertical line is a line which connects two points whose x-coordinates are same. Two horizontal lines of size 1 can get connected to form a horizontal line of size 2. For eg, the line connecting the points (7,1) & (12, 1) and the line connecting the points (12,1) & (27,1) can join together to form a horizontal line of size 2, with the endpoints as (7,1) and (27,1).

Given the coordinates of n points , write an algorithm and the pseudocode to compute the the maximum size of the horizontal line and maximum size of the vertical line that can be formed with the given points.

**Input format**

Enter the number of points to be entered : n

Next ‘n’ lines contain the x and y coordinates of the lines separated by a space

**Output format**

Print maximum size of the horizontal lines that can be formed with the given n-points

Print maximum size of the vertical line that can be formed with the given n-points

**Illustration :**

5

0 2

0 1

2 0

1 0

0 0

Output

2

2

Note the points entered are : (0,2), (0,1), (2,0),(1,0) and (0,0).

A vertical line connecting the vertical with the endpoints (0,0)& (0,1) and a vertical line with the endpoints (0,1) & (0,2). The size of this vertical line is 2.

Hence maximum size of the vertical line that can be formed with the given points is 2.

**ANS.**

**from collections import Counter**

**n,coor1,coor2 = int(input()),[],[]**

**for i in range(n):**

**x,y = map(int,input().split())**

**coor1.append(x)**

**coor2.append(y)**

**z1 = Counter(coor1).most\_common(1)[0][0]**

**z2 = Counter(coor2).most\_common(1)[0][0]**

**hor,ver = 0,0**

**for i in range(len(coor1)):**

**if coor1[i] == z1: ver+=1**

**for i in range(len(coor2)):**

**if coor2[i] == z2: hor+=1**

**print(hor-1,ver-1,sep="\n")**

**OPTIMISED:**

**from collections import Counter**

**n,coor1,coor2 = int(input()),[],[]**

**for i in range(n):**

**x,y = map(int,input().split())**

**coor1.append(x); coor2.append(y)**

**z1 = Counter(coor1).most\_common(1)[0]**

**z2 = Counter(coor2).most\_common(1)[0]**

**hor,ver = z2[1],z1[1]**

**print(hor-1,ver-1,sep="\n")**

#### **Digit Mapping for Characters**

Given a string S with distinct letter find a mapping for each letter in it such that the number formed by replacing the letter with the numbers is a largest possible perfect square with different digits.

A number is said to be perfect square if the number is square of an integer. For example, if the word given is ‘care’ then the following map shall be done for the characters:

c – 9

a – 8

r – 0

e – 1

The number 9801 is a perfect square since 992 is 9801.

**Input Format**

First line contains the string for which numbers are to be assigned for characters

**Output Format**

Print character of the string and the number mapped for it separated by a space

**ANS.**

**s = input()**

**l = len(s)**

**num = 10\*\*l-1**

**ans = 0**

**while(len(str(num)) > l-1):**

**num-=1**

**if int(num\*\*0.5) == num\*\*0.5 and len(list(set(list(str(num))))) == len(str(num)):**

**ans = num**

**break**

**info = dict(zip(list(s),list(str(ans))))**

**for i in info:**

**print(i,info[i])**

#### **Moving Averages**

Consider n numbers. Average of the given n numbers is the sum of all the numbers divided by n. Moving average of size k, k<=n, written as Mvk , is the average of the first k numbers among the n numbers. For a number with n digits, there will be n moving averages. Grand Moving average is the average of all the possible moving averages. If we have the numbers 1,7,8,9; moving average of size 1 is 1. Moving average of size 2 is (1+7)/2= 4. Moving average of size 3 is (1+7+8)/3=5.33; Moving average of size 4 is (1+7+8+9)/4 = 6.25. Grand moving average is (1+4+5.33+6.25)/4= 16.58/4=4.14

Given n numbers, Write the pseudocode and the code to compute the Grand Moving Average of the given numbers. The grand moving average should be printed in two decimal format.

**Input format**

First line contains the number of numbers, n

Next ‘n’ lines contain the numbers in the collection

**Output format:**

Print grand Moving Averages in two decimal format

**Illustration :**

**Input**

4

1

7

8

9

**Output**

4.14

**ANS.**

**#include <iostream>**

**#include <iomanip>**

**using namespace std;**

**int main(){**

**int n,sum = 0;**

**double sum1 = 0;**

**cin>>n;**

**for(int i=0;i<n;i++){**

**int a;**

**cin>>a;**

**sum += a;**

**sum1 += sum/(i+1);**

**}**

**cout<<fixed<<setprecision(2)<<sum1/n;**

**}**

**PASCAL’S TRIANGLE (O(N^2) time complexity):**

**#include <iostream>**

**#include <vector>**

**#include <cstdlib>**

**using namespace std;**

**int main(){**

**int n;**

**cin>>n;**

**vector <vector<int>> pascal = {{1},{1,1}};**

**if(n == 0){**

**cout<<"empty";**

**exit(0);**

**}**

**else if(n == 1){**

**cout<<1<<endl;**

**exit(0);**

**}**

**else{**

**for(int i=2;i<n;i++){**

**int prev = i-1;**

**vector <int> step = {1};**

**for(int j=0;j<pascal[prev].size()-1;j++){**

**step.push\_back(pascal[prev][j]+pascal[prev][j+1]);**

**}**

**step.push\_back(1);**

**pascal.push\_back(step);**

**}**

**}**

**for(auto x: pascal){**

**for(auto y: x)**

**cout<<y<<" ";**

**cout<<"\n";**

**}**

**}**

#### **String generation using W-function**

Using an N-function, every letter of English alphabet (either upper case or lower case) can be represented as a number that corresponds to its position in the English alphabet. For example, ‘A’ is represented by 1, ‘z’ by 26. This is written as N(A)=1, N(a)=1. In the same way, using a W-function, a number k in the range, from 1 to 26 (both included) , is represented by the symbol whose positional index in the English alphabet, is k. This is written as W(25)=y.

From an integer n, two strings W1 and W2 may be generated in such a way that every character ‘c’ in W1 or W2, taken in order, is equal to W(k)=c where k is a single digit for generating W1 and it is a two digit number for generating W2, that occurs in order in the given integer ‘n’. When the two digit number, k is greater than 26 take the letter corresponding to modulus 26 of k.

Given an integer ‘n’ which does not have any zero as its digits, write an algorithm and the pseudocode to generate the two strings that can be formed from the number n using W-function.

Print both the words in lower case letters

For example, if n = 1234 then words are generated from n as follows.

1-2-3- 4 generates abcd

12-23-34 generates lwh

When n= 491

4-9-1 generates dia

49-91 – generates wm

**Input format :**

First line contains a number, n

**Output format**

Print W1 in first line

Print W2 in second line

**ANS.**

**C++**

**#include <iostream>**

**#include <string>**

**#include <map>**

**using namespace std;**

**int main(){**

**map <int,char> alpha;**

**for(int i=1;i<=26;i++) alpha[i] = (char)int('a')+i-1;**

**string num;**

**cin>>num;**

**string w1 = "",w2 = "";**

**for(int i=0;i<num.length();i++){**

**int dig = int(num[i]-'0');**

**if(dig <= 26) w1+=alpha[dig];**

**if(dig > 26) w1+=alpha[dig%26];**

**if(i<=num.length()-2){**

**char w[2];**

**w[0] = num[i];**

**w[1] = num[i+1];**

**int num1 = atoi(w);**

**if(num1 <= 26) w2+=alpha[num1];**

**if(num1 > 26) w2+=alpha[num1%26];**

**}**

**}**

**cout<<w1<<"\n"<<w2;**

**}**

**PYTHON**

**alpha = {}**

**for i in range(1,27): alpha[i] = chr(ord('a')+i-1)**

**num,w1,w2 = input(),"",""**

**for i in range(len(num)):**

**if int(num[i])<=26: w1+=alpha[int(num[i])]**

**else: w1+=alpha[int(num[i])%26]**

**if i <= len(num)-2:**

**if int(num[i]+num[i+1]) <= 26: w2+=alpha[int(num[i]+num[i+1])]**

**else: w2+=alpha[int(num[i]+num[i+1])%26]**

**print(w1,w2,sep="\n")**

#### **Game of Dragons**

IronMan is stuck on a level of the Infinity War game which he is playing now. To advance this level, he has to kill all the **N** dragons that are present on this level. IronMan and the dragons have strength, which is represented by an integer. In the fight between him and dragon, the fight's outcome is determined by their strength. Initially, IronMan's strength equals **S**.

If IronMan starts dueling with the **ith (1 <= i <= N)** dragon and IronMan's strength is not greater than the dragon's strength **Xi**, then IronMan loses the duel and dies. But if IronMan's strength is greater than the dragon's strength, then he defeats the dragon and gets a bonus strength increase by **Yi**.

**IronMan can fight the dragons in any order**. Your task is to find out if he will be able to win all the duels without any loss.

**Input Format:**  
The first line will contain two space-separated integers, the initial strength S and the number of dragons N.  
Then N lines follow, each of which contains two space-separated integers **Xi** and **Yi**, where **Xi** is the strength of the dragon and **Yi** is the bonus strength which IronMan gains after defeating this dragon.

**Output Format:**  
Print a single line containing either "YES" (without quotes), if IronMan can defeat all dragons or "NO" (without quotes) if he can't.

**For example:**  
**Input**:  
2 2  
1 99  
100 0

**Output**:  
YES

**Explanation**:  
IronMan's initial strength is 2 and the number of dragons also equals 2  
The first dragon has strength equal to 1 and thus IronMan can defeat this dragon and gain 99 points  
making his strength equal to 2 + 99 = 101.  
Since 101 > 100, IronMan can defeat this dragon also and thus the answer is YES.

**Second example:**

**Input:**  
8 1  
100 100

**Output:**  
NO

**Explanation**:  
IronMan's initial strength is 8 but the required strength for the first dragon is 100 and hence, he dies and we print "NO".

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**#define long long int;**

**using namespace std;**

**bool comp(pair<int,int> x,pair <int,int> y){**

**if(x.first == y.first) return(x.second < y.second);**

**else return(x.first < y.first);**

**}**

**int32\_t main(){**

**int s,n;**

**vector <pair<int,int>> dragdata;**

**cin>>s>>n;**

**for(int i=0;i<n;i++){**

**pair <int,int> x;**

**cin>>x.first>>x.second;**

**dragdata.push\_back(x);**

**}**

**sort(dragdata.begin(),dragdata.end(),comp);**

**bool flag = true;**

**for(auto y: dragdata){**

**if(y.first < s) s += y.second;**

**else{**

**flag = false;**

**break;**

**}**

**}**

**if(flag == true) cout<<"YES";**

**else cout<<"NO";**

**}**

**PYTHON:**

**s,n = map(int,input().split())**

**dragon = []**

**for i in range(n):**

**x = tuple(map(int,input().split()))**

**dragon.append(x)**

**dragon.sort()**

**flag = 0**

**for i in dragon:**

**if i[0] < s: s+=i[1]**

**else: flag = 1; break**

**if flag == 1: print("NO")**

**else: print("YES")**

#### **Rotating Words**

After solving the previous problem, Kartik is stuck on yet another new problem given to him by his teacher.  
The problem wants him to find out the number of unique words that can be formed by cyclically rotating a given string.

Cyclic rotation of a string is defined as moving the last character of the string on the first position.

Kartik can apply Cyclic rotation any number of times.

You have to help him in finding the maximum number of unique words that can be formed by applying Cyclic rotation any number of times.

**Input Format:**  
The only line of the input contains a string S containing of lowercase characters

**Output Format:**  
Print only one single integer denoting the number of unique words that can be formed as explained above.

**Constraints:**  
1 <= len(s) <= 50

**Example:**  
**Input:**  
abcd  
**Output:**  
4  
**Explanation:**  
abcd,dabc,cdab,bcda  
4 unique strings are possible

**Input:**  
cc  
**Output:**  
1  
**Explanation:**  
Only 1 unique string is possible , 'cc'

**ANS.**

**from collections import deque**

**s = deque(input())**

**words,count = [],0**

**for i in range(len(s)):**

**s.appendleft(s.pop())**

**if "".join(s) not in words: count+=1; words.append("".join(s))**

**print(count)**

#### **Fibanocci Increment Mixed String**

Increment mixed string is an operation which operates on two strings S1 and S2 of same length to generate a new string S3. The letters in odd position of S3 is one more than the corressponding letter in S1 and the letters in even position of S3 is one more than the corressponding letter in S2. For example, if S1 = ‘amey’ and S2 = ‘dhft’ then S3 = ‘bifu’. For letter ‘z’, letter ‘a’ is one more than it.

Fibanocci increment mixed string is operation which operates on the last two strings in the series. Given two strings, S1 and S2 write a code to generate the nth element using Fibanocci increment mixed string operation. The given strings S1 and S2 are the first two elements in the Fibanocci increment mixed string series. Third element in the series is found by applying increment mixed string operation for first two elements.

If S1 = ‘amey’ and S2 = ‘dhft’ then the first five elements in the series are as follows:

amey

dhft

bifu

ejgv

ckgw

**Input Format**

First line contains the string S1

Next line contains the string S2

Next line contains the value of n

**Output Format**

Print the nth element in the Fibanocci increment mixed string series

**ANS.**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**int main(){**

**int n;**

**string s1,s2,s3 = "";**

**cin>>s1>>s2>>n;**

**for(int k=0;k<n-2;k++){**

**int j = -1;**

**for(int i=0;i<s2.length();i++){**

**if(i%2 == 0){**

**if(s1[++j] != 'z') s3+=(char)((int)s1[j])+1;**

**else s3+='a';**

**}**

**else{**

**if(s2[++j] != 'z') s3+=(char)((int)s2[j])+1;**

**else s3+='a';**

**}**

**}**

**s1 = s2;**

**s2 = s3;**

**s3 = "";**

**}**

**cout<<s2;**

**}**

#### **Image Transformation**

Digital images are represented as a matrix and each element in the matrix represents the RGB value of a pixel. Given the pixel values of a nXn image. Write a code to transform the image into another image by alternate swap of edge pixels. That is for example, given a 4 X 4 image as follows:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

The edge elements of the above image are 1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9 and 5. The following steps are involved in the transformation:

1 and 3 are swapped

1 and 8 are swapped

1 and 16 are swapped

1 and 14 are swapped

1 and 9 are swapped

After doing alternate swap of edge elements the pixel values of the image looks as follows:

3 2 8 4

5 6 7 16

1 10 11 12

13 9 15 14

And given an image of dimension 5 X 5 as follows:

2 4 5 6 3

3 11 12 7 9

7 15 13 8 1

2 21 25 13 14

17 10 16 19 3

Should transform as:

5 4 3 6 1

3 11 12 7 9

2 15 13 8 3

2 21 25 13 14

7 10 17 19 16

**Input Format**

First line contains the dimension of the matrix, n

Next nxn lines contain the elements of the matrix in row major order. That is the elements in the first row are given followed by second row and so on

**Output Format**

Print the transformed nxn matrix in ‘n’ lines

Each line contains the elements of a row separated by a tab.

**Note:**There is a tab at the end of each line

**ANS.**

**n,matrix = int(input()),[]**

**for i in range(n):**

**row = []**

**for j in range(n):**

**row.append(int(input()))**

**matrix.append(row)**

**frame = []**

**for j in range(n):**

**frame.append(matrix[0][j])**

**for j in range(1,n):**

**frame.append(matrix[j][n-1])**

**for j in range(n-2,-1,-1):**

**frame.append(matrix[n-1][j])**

**for j in range(n-2,0,-1):**

**frame.append(matrix[j][0])**

**for i in range(0,len(frame),2):**

**try: frame[i],frame[i+2] = frame[i+2],frame[i]**

**except: break**

**for k in range(n):**

**matrix[0][k] = frame.pop(0)**

**for k in range(1,n):**

**matrix[k][n-1] = frame.pop(0)**

**for k in range(n-2,-1,-1):**

**matrix[n-1][k] = frame.pop(0)**

**for k in range(n-2,0,-1):**

**matrix[k][0] = frame.pop(0)**

**for i in range(n):**

**for j in range(n):**

**print(matrix[i][j],end = "\t")**

**print()**

#### **Length of Closed Figures**

Two lines P1 and P2 are said to be connected if the end point of P1 and start point of P2 are same. In this problem, there is a collection of points in a two dimensional space and the point numbers are given to represent a line. All the lines have distinct starting point and ending point. Given the start and end point numbers of ‘n’ lines and a check point number ‘p’ write a code to find out the length of the closed figure that can be formed that starts with the check point number ‘p’. For example, given ten pairs of point numbers for 10 lines as follows:

(2, 5)

(7, 11)

(13, 14)

(11, 15)

(17, 18)

(15, 17)

(18, 25)

(32, 40)

(25, 32)

(40, 7)

and a check point number as 7, a closed figure of length 8 can be formed with the points:

(7, 11) - (11, 15) - (15, 17) - (17, 18) - (18, 25) - (25, 32) - (32, 40) - (40, 7) - (7, 11)

**Input Format**

First line contains the number of lines, n

Next ‘n’ lines contain the number of the start and end points

Next line contains the check point, p

**Output Format**

Length of the closed figure that can be formed with check point, p

Print zero if a closed figure cannot be formed with check point, p

**ANS.**

**n = int(input())**

**graph = {}**

**for i in range(n):**

**u,v = map(int,input().split())**

**graph[u] = v**

**p = int(input())**

**count = 0**

**def trav(graph,u,p,count):**

**try:**

**if graph[u] == p:**

**count+=1**

**print(count)**

**return**

**count+=1**

**trav(graph,graph[u],p,count)**

**except:**

**print(0)**

**return**

**trav(graph,p,p,count)**

#### **Find the Lucky Kids**

In a game for kids under age seven, m\*n kids are made to sit in a room where there are ‘n’ chairs arranged in ‘m’ rows. The game organizers has got ‘m\*n’ chits in which either one of the two numbers ‘p’ or ‘q’ is written. Each kid is asked to pick a chit. A lot is taken by the coordinator which contains the lucky number ‘l’ (Value of 'l' will be either 'p' or 'q'). Then it is declared that “Chit Transition” takes place for few kids. If the lucky number ‘l’ is equal to ‘p’, during chit transition, a kid ‘k’, sitting in the row ‘r’ and column ‘c’ with number ‘p’ is given ‘q’ when one of the following criteria is satisfied:

(i) If atleast one of the kid sitting in the row ‘r’ has ‘q’

(ii) If atleast one of the kid sitting in the column ‘c’ has ‘q’

After this Chit transition phase, the kids with number ‘p’ are declared as winners. For example, if there are 3\*3 kids, chit numbers as 3 and 5 seated with original chit number as follows:

3 5 3

3 3 3

3 5 5

If the lucky number taken by coordinator is 3 then the chit number of the kids after “Chit Transition” phase will be:

5 5 5

3 5 5

5 5 5

Now the kid sitting in the second row and first column is the winner. Sometimes there can be more than one winner in this game and sometimes there may not be any winner in this game also. Print the row and column position of all the winning kids and when there are no winners print ‘No winner’.

**Input Format**

First line contains the number, m

Next line contains the number, n

Next line contain the number, p

Next line contain the number, q

Next ‘m’ lines contain the chit number of ‘n’ kids seated on each row separated by a space

Next line contains the lucky number, l

**Output Format**

Print the row and column position of winning kids in order. Print the position sorted row wise, when row of more than one position is same print them sorted column wise and Print ‘No winner’ when no one has won the game

**ANS.**

**m,n,p,q = int(input()),int(input()),int(input()),int(input())**

**seating = [list(map(int,input().split())) for i in range(m)]**

**l,transpose = int(input()),[]**

**for i in range(n):**

**row = []**

**for j in range(m):**

**row.append(seating[j][i])**

**transpose.append(row)**

**winners = []**

**for i in range(m):**

**for j in range(n):**

**if set(seating[i]) == set(transpose[j]) == {l}: winners.append((i+1,j+1))**

**if len(winners) == 0: print("No winner")**

**else:**

**for i in winners: print(\*i)**

#### **Throwball Game**

A company is organizing a game for its employees on the occassion of new year celebrations. The game is conducted by a coordinator and has a group of ‘n’ players. In this game, all the players of the game are assured a prize money. The players are made to stand together one side of the net and as a first step the coordinator throws a ball towards them from the other side. The player ‘p’ who catches the ball first gets a prize money of ‘m’. Then the player ‘p’ is asked to go to the other side and throw the ball towards the rest of the players. This chain of throwing the ball continues such that the person who caught the ball in ith throw, throws the ball towards rest of the players in the i+1th throw. The chain continues till there is only one player left out in the group. The prize money for the players gets reduced by 2% for each catch. Given the maximum prize money ‘m’, name of the players in ascending order and the index of the name of the player who caught the ball when it was thrown by that particular player, name of the player who caught the ball in the frist throw, write a code to find out the prize money won by a player ‘x’.

For example, when there are seven players, maximum prize money is 5000, ball was caught by player at index 6 in the first thrown, and details of the players as follows, the prize money won by player A is 94% of 5000 and it is 4700.00

|  |  |  |
| --- | --- | --- |
| **Index** | **Name of the Player** | **Index of player who caught the ball** |
| 1 | A | 3 |
| 2 | B | 5 |
| 3 | C | 2 |
| 4 | D | 7 |
| 5 | E | 0 |
| 6 | F | 4 |
| 7 | G | 1 |

The above table indicates that when the ball was thrown by the player ‘A’ it was caught by the player whose name is at index 3, when the ball was thrown by the player ‘B’ it was caught by the player whose name is at index 5 and so on. The player who caught the ball last has a 0 as the index of the player who caught the ball when it was thrown by that player.

**Input Format**

First line contains the maximum prize amount, m

Next line contains the number of players, n

Next ‘n’ lines contain the name of th players in ascending order and the index of name of the player to whom each player threw the ball, separated by a space

Next line contains the index of the name of the player who caught the ball first,

Next line contains the name of the player whose prize money has to be found, x

**Output Format**

Print the prize money won by the player, x. Print only two decimal places

**Example:**

**Input:**

5000

7

A 3

B 5

C 2

D 7

E 0

F 4

G 1

6

A

**Output:**

6500.00

**ANS.**

**m,n = int(input()),int(input())**

**index\_info,info = {},{}**

**for i in range(n):**

**a,b = map(str,input().split())**

**b = int(b)**

**info[i+1] = b**

**index\_info[i+1] = a**

**f,x = int(input()),input()**

**wins = []**

**def order(info,person\_throwing,x):**

**if person\_throwing == 0:**

**for i in info:**

**if info[i] == 0: wins.append(i); break**

**return**

**if index\_info[info[person\_throwing]] == x:**

**wins.append(index\_info[person\_throwing])**

**wins.append(x)**

**return**

**wins.append(index\_info[person\_throwing])**

**order(info,info[person\_throwing],x)**

**order(info,f,x)**

**prize = m**

**print(format(prize-wins.index(x)\*2\*m/100,".2f"))**

#### **People and Distances**

You are given the top view of a ground, where there are two values: '0' and '1'. Each '1' denotes that there is a person standing at that   
position in the ground. Recently Raman was asked to find the number of distinct pairs of people who have given manhattan distances between them.  
Raman doesn't like to disappoint and hence he has come to you for help.

For more information on Manhattan distance, see [here](https://xlinux.nist.gov/dads/HTML/manhattanDistance.html).

**Input Format:**  
The first line of the input contains two space-separated integers 'n' and 'm' denoting the number of rows and number of columns in the ground.  
Then 'n' lines follow, each of which contains a string containing '0's and '1's.  
The next line contains the value of 'q' denoting the number of queries.  
Each of the following lines contain one integer denoting the distance

**Output Format:**  
Print q lines, each containing the number of distinct pairs of people who have manhattan distance equal to the given manhattan distance.

**It is guaranteed that the distance given in query will be between 1 and n+m-2 (both inclusive).**

**Example:**  
**Input:**  
3 4  
0001  
0100  
0010  
5  
1  
2  
3  
4  
5

**Output:**  
0  
1  
2  
0  
0

**ANS.**

**n,m = map(int,input().split())**

**ground = []**

**for i in range(n):**

**row = list(input())**

**ground.append(row)**

**tovisit = []**

**for i in range(n):**

**for j in range(m):**

**if ground[i][j] == '1': tovisit.append((i+1,j+1))**

**distances = []**

**for i in range(len(tovisit)):**

**for j in range(i+1,len(tovisit)):**

**dist = abs(tovisit[i][0]-tovisit[j][0])+abs(tovisit[i][1]-tovisit[j][1])**

**distances.append(dist)**

**for q in range(int(input())):**

**query = int(input())**

**if query in distances: print(distances.count(query))**

**else: print(0)**

#### **Smallest Divisible Number**

Given an integer ‘n’ where n>=1, write a C code to check whether there exist a smallest number ‘S’, where S<=107 and S is divisible by all even numbers ‘m’ where 1<=m<=n.

For example, when n = 10, the smallest number divisible by all even numbers 2, 4, 6, 8 and 10 is 120. If no such number exist then print ‘No such number in range’.

**Input Format**

First line contains the number, n

**Output Format**

Print either the number that is divisible by all even numbers less than ‘n’ or print No such number in range

**ANS.**

**def gcd(a,b):**

**if b == 0: return a**

**else: return gcd(b,a%b)**

**def lcm(a,b):**

**return (a\*b)/gcd(a,b)**

**n = int(input())**

**even,lcm\_even = [i for i in range(2,n+1) if i%2 == 0],1**

**for i in even: lcm\_even = lcm(lcm\_even,i)**

**if lcm\_even <= 10\*\*7: print(int(lcm\_even))**

**else: print("No such number in range")**

#### **Principal Numbers**

A proper factor of a number is a factor of the number other than itself. Given a collection of numbers, a number n1 in position ‘i’ is said to be principal number if the largest proper factor of all numbers in position > i is lesser than or equal to the largest proper factor of n1. The last number in the collection is a principal number. For example, given five numbers 29, 15, 5, 9 and 11, the output should be 15, 9 and 11.

**Input Format**

First line contains the number of elements, n

Next ‘n’ lines contain the elements in the collection

**Output Format**

Print the principal numbers one on each line

**ANS.**

**FIRST SOLN.(brute force more efficient):**

**n = int(input())**

**l = [int(input()) for i in range(n)]**

**factors = []**

**for i in l:**

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

**if i%j == 0: factors.append(j); break**

**for i in range(n):**

**flag = 0**

**for num in factors[i+1:]:**

**if num > factors[i]: flag = 1; break**

**if flag == 0: print(l[i])**

**SECOND SOLN. (less efficient but better approach, could be optimized):**

**n = int(input())**

**l = [int(input()) for i in range(n)]**

**factors,ans = [],[]**

**for i in l[::-1]:**

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

**if i%j == 0: factors.append(j); break**

**flag = 0**

**for k in factors:**

**if k > factors[-1]: flag = 1; break**

**if flag == 0: ans.append(i)**

**for i in ans[::-1]: print(i)**

#### **Skipping stones**

Rahul is playing a game known as skipping stones. In this game, in order to reach the end he can either jump past 1 stone or 2 stones at a time. He enjoyed playing the game but   
wonders in how many ways can he reach to the Nth step. Please help Rahul in finding the answer to this problem.

**Input Format:**  
The first line containing Q, denoting the number of queries to answered.  
Each of the next Qi, lines denote the value of Nth stone.

**Output:**  
Print Q lines, each containing the value of ith query on a single line. If the answer is greater than 32 bit integer output it modulo 10^9+7.

**Constraints:**  
1 <= Q <= 1000  
The value of N in each query can go upto 1000.

**Example:**  
**Input:**  
2  
1  
4  
**Output:**  
1  
5

**Explanation:**  
There are 5 ways in which Rahul can reach to the 4th stone, these are:  
1. Going 1 by one to each stone: (1,1,1,1)  
2. (2,2)  
3. (1,2,1)  
4. (2,1,1)  
5. (1,1,2)

**ANS.**

**DRAW FOR VARIOUS TEST CASES, WORKS LIKE FIBONACCI ONLY**

**dp = [-1 for i in range(1001)]**

**def solver(n):**

**if dp[n] != -1: return dp[n]**

**else:**

**if n == 1: dp[n] = 1; return dp[n]**

**if n == 2: dp[n] = 2; return dp[n]**

**dp[n] = (solver(n-1)+solver(n-2))%(10\*\*9+7)**

**return dp[n]**

**for i in range(int(input())):**

**query = int(input())**

**print(solver(query))**

**ITERATIVE DP SOLUTION:**

**q = int(input())**

**queries = [int(input()) for i in range(q)]**

**ans = [1,2]**

**for i in range(max(queries)-2): ans.append((ans[-1]+ans[-2])%(10\*\*9+7))**

**for query in queries: print(ans[query-1])**

#### **Time Taken to Clarify Doubts**

A maths teacher wants to make all his students to have a good understanding of the concepts. He regularly give homeworks and clarifies the doubts of his students. Every day some ‘n’ number of students come to him with some number of their doubts to the teacher at the same time. The teacher do not want any of his student to wait for a long time without attended since that may lead them to get frustrated and lose interest. So he follows a particular statergy to clarify their doubts in which the teacher takes multiple rounds to clarify the doubts of a student till all his doubts are clarified. The steps involved in his startegy are:

(i) Students with doubts are made to stand in a queue

(ii) In each round, the teachers clarifies ‘t’ doubts for each student. He starts from the first student in the queue and move till the last student.

(iii) A student leaves the queue as soon as all of his doubts got clarified. Assume that one minute will be taken by the teacher to clarify one doubt.

Given the number of students coming to clarify their doubts, the number of doubts each student has got and the number of doubts ‘t’ that are going to be clarified by the teacher in each round, write a C program to find the number of minutes for each student to clarify his doubts and leaves the queue.

For example, when there are three students with number of doubts as 5, 7 and 8 and the teacher decides to clarify two questions for a student in a round. Time taken for each student to clarify doubts is 13, 18 and 20.

**Input Format**

First line contains the number of students, n

Next ‘n’ lines contain the number of doubts each student needs to clarify

**Output Format**

Print the time taken for each student to clarify his doubts and leave the queue

**ANS.**

**n = int(input())**

**class1,total\_doubts = [],0**

**for i in range(n): class1.append(int(input())); total\_doubts+=class1[i]**

**t = int(input())**

**solve\_time = [-1 for i in range(n)]**

**time = 0**

**class1\_flag = [True]\*n**

**while time < total\_doubts:**

**for i in range(n):**

**if class1[i] > t: class1[i]-=t; time+=t**

**else: time+=class1[i]; class1[i] = 0**

**if class1[i] == 0 and class1\_flag[i]:**

**solve\_time[i] = time; class1\_flag[i] = False**

**for i in range(n): print(solve\_time[i])**

#### **Print the Pattern**

Given the value of ‘n’ write a C program to print a special rectangular pattern with dots and starts. When the value of n is 5 the rectangle looks as below:

. . . . \* . . . .

. . . \* . \* . . .

. . \* . . . \* . .

. \* . . . . . \* .

\* . . . . . . . \*

**Input Format**

First line contains the value of ‘n’

**Output Format**

Print the pattern appropriate to the value of ‘n’

Dots and stars are separated by one space and each row in the pattern is separated only by a new line character.

**ANS.**

**PYTHON:**

**n = int(input())**

**k = 0**

**for i in range(n):**

**for j in range(2\*n-1):**

**if j == n-1-k or j == n-1+k:**

**if j != 2\*n-2: print("\*",end = " ")**

**else: print("\*")**

**else:**

**if j != 2\*n-2: print(".",end = " ")**

**else: print(".")**

**k+=1**

**C++:**

**#include <iostream>**

**using namespace std;**

**int main(){**

**int n,k=0; cin>>n;**

**for(int i=0;i<n;i++){**

**for(int j=0;j<2\*n-1;j++){**

**if(j == n-1-k || j == n-1+k){**

**if(j != 2\*n-2) cout<<"\*"<<" ";**

**else cout<<"\*"<<endl;**

**}**

**else{**

**if(j != 2\*n-2) cout<<"."<<" ";**

**else cout<<"."<<endl;**

**}**

**}**

**k++;**

**}**

#### **Minimum Number of Bits Required**

Memory usage has to be minimized in an embedded system application and the embedded system uses same number of bits to store all numbers of the application. The programmer wants to use only the minimum number of bits required for storing ‘n’ integers. Given a set of ‘n’ integers, write a C++ program to find the minimum number of bits required to store each of the integer. For example, if five integers 4, 16, 31, 23, 2 are to stored then the minimum number of bits required to store each integer is 5.

**Boundary Conditions**

0<=x<=922337203685477580 where ‘x’ is any value in the number

**Input Format**

First line contains the number of numbers used in the application, n

Next line contains the numbers of the application separated by a space

**Output Format**

Print the minimum number of bits required to store the numbers of the application

**ANS.**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**#define ll long long**

**int main(){**

**int n; cin>>n;**

**ll max = 0;**

**for(int i=0;i<n;i++){**

**ll ele; cin>>ele;**

**if(max < ele) max = ele;**

**}**

**if(pow(2,log2(max)) == max) cout<<(int)log2(max);**

**else cout<<(int)log2(max)+1;**

**}**

#### **Form Names with Tiles**

Tiles with alphabets are arranged in a linear fashion and each tile in the arrangement has an index ‘i’.

A child is given ‘n’ tiles and asked to check how many times he can form his name by rearranging the tiles. But there is a constraint, to form his name once the child can rearrange tiles from any position ‘p’ to position ‘p+q-1’ where q is the length of the name of the child. For example, if the tiles are arranged are as ‘abacdefhjcabdaabefrt’ and the child’s name is ‘abcd’ then the child can form his name by using tiles from position ‘2’ and using tiles from position ‘10’.

**Input Format**

First line contains the letters in the tile

Next line contains the name of the child

**Output Format**

Print the position of the tiles from where the child’s name can be formed

**ANS.**

**#include <iostream>**

**#include <map>**

**using namespace std;**

**int main(){**

**string n,l;**

**cin>>l>>n;**

**map<char,int> f;**

**int len = n.length();**

**for(int i=0;i<len;i++){**

**if(f.find(n[i]) == f.end()) f[n[i]]=1;**

**else f[n[i]]++;**

**}**

**for(int i=0;i<l.length();i++){**

**map<char,int> s;**

**bool flag = true;**

**for(int j=0;j<len;j++){**

**if(f.find(l[i+j]) != f.end()){**

**if(s.find(l[i+j]) == s.end()) s[l[i+j]] = 1;**

**else if(s[l[i+j]] < f[l[i+j]]) s[l[i+j]]++;**

**else {flag = false; break;}**

**}**

**else{**

**flag = false;**

**break;**

**}**

**}**

**if(flag){**

**for(auto x: f){**

**if(x.second != s[x.first]){**

**flag = false;**

**break;**

**}**

**}**

**if(flag) cout<<i+1<<"\n";**

**}**

**}**

**}**

#### **Product of Two 2-Digit with ‘+’ and ‘/’**

Based on the yesterday’s problem told by Karthik, Vijay got curious to know more about tricks by Shakuntala Devi and he started reading her books. He liked the trick which describes a method to find product of two 2-digit numbers with ‘+’ and ‘/’ operators only. Given two 2-digit numbers ‘m’ and ‘n’,

(i) Write the lesser number ‘l’ on left side of a paper

(ii) Divide ‘l’ by 2 and write quotient on one line ignoring remainders till 1 is reached.

(iii) Write the larger number on the right side of the paper in line number 1

(iv) Then for right side of each line, write double of the value in previous line.

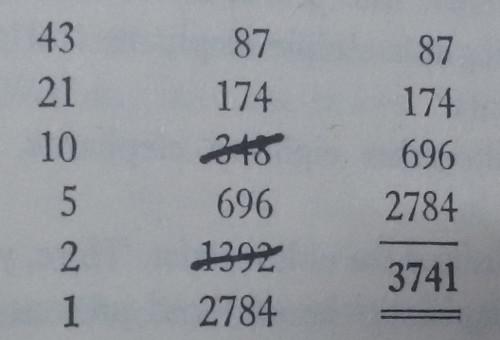
(v) Strike out the lines of even quotients in left side.

(vi) Sum up the remaining values on the right side

This gives product of m and n.

Given two 2-digit numbers, write a C++ program to find product of the numbers using above procedure and print the values on the left and right sides of the papers that are not stroked out in ascending order of left side value.

For example, if the numbers are 43 and 87 then the procedure works as follows:

****

And the expected output is:

1 2784

5 696

21 174

43 87

3741

**Input Format**

First line contains the first two-digit number, m

Next line contains the second two-digit number, n

**Output Format**

Print all the lines that are not stroked out in the above procedure

Print the product of m and n

**ANS.**

**#include <iostream>**

**#include <cmath>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**int m,n; cin>>m>>n;**

**int larg = max(m,n),smal = min(m,n);**

**int len = pow(2,log2(smal))==smal?(int)log2(smal)+1:(int)log2(smal);**

**int i=0,ans=0,larr[len],sarr[len];**

**while(smal >= 1){**

**if(smal%2){**

**sarr[i] = smal;**

**larr[i] = larg;**

**ans+=larg;**

**i++;**

**}**

**smal/=2; larg\*=2;**

**}**

**for(int j=i-1;j>-1;j--){**

**cout<<sarr[j]<<" "<<larr[j]<<"\n";**

**}**

**cout<<ans;**

**}**

#### **Generate Letters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 10 | 9 | 8 | 7 | 6 |

In a game a string S is given to child and the child is asked to check if it can put each letter of the string in sequence in the boxes arranged in two rows so that the letters in the boxes placed in the same column are same.

That is if there are 10 characters in the string then the child can place them in the order shown above. If the characters in the string are not in such a way that this can be done then new characters are to be added to the string at the end to achieve this.

For example, if the string is ab then string is expanded to abba to achieve this. The number of boxes needs to be minimum and no character can be removed from the string.

**Input Format**

First line contains the string, s

**Output Format**

First line contains the string that can achieve the arrangement

**ANS.**

**#include <iostream>**

**#include <string.h>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**string s;**

**cin>>s;**

**int n = s.length(),flag = 0;**

**for(int i=0;i<n/2;i++){**

**if(s[i] != s[n-i-1]) {flag = 1; break;}**

**}**

**if(!flag) cout<<s;**

**else{**

**string rev = "";**

**int max\_rev\_pal = n-1;**

**for(int i=n-1;i>=0;i--){**

**rev+=s[i];**

**int flag1 = 0;**

**for(int j=0;j<=(n-i)/2;j++){**

**if(rev[j] != rev[(n-i)-1-j]){**

**flag1 = 1; break;**

**}**

**}**

**if(!flag1) max\_rev\_pal = i;**

**}**

**if(max\_rev\_pal == n-1) cout<<s+rev;**

**else{**

**string add = s.substr(0,max\_rev\_pal);**

**for(int i=0;i<max\_rev\_pal/2;i++){**

**swap(add[i],add[max\_rev\_pal-1-i]);**

**}**

**cout<<s+add;**

**}**

**}**

**}**

#### **Nullify Bits**

For a given integer, odd\_bit\_value is defined as the number of bits which are '1' in odd position and even\_bit\_value is defined as the number of bits which are '1' in even position. Nullification is a process in which the 1's in odd position are made as zero if odd\_bit\_value is lesser than even\_bit\_value and it makes 1's in even position as zeros otherwise.

In this process, number of bits to be considered for this process is equal to the minimum number of bits required to represent the number. The right most bit of the number is considered to at poistion 1. For example, if n is 12 - 1100, it becomes 4 and 15 becomes 5.

Given an integer, write a C++ program to print the number after nullification.

**Input Format**

First line contains number, n

**Output Format**

Print the number after nullification

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <cmath>**

**using namespace std;**

**int main(){**

**ios\_base::sync\_with\_stdio(false);**

**cin.tie(NULL);**

**int n; cin>>n;**

**vector<int> bin;**

**int i=1,odd=0,even=0;**

**while(n > 0){**

**bin.push\_back(n%2);**

**if(i%2 && n%2) odd++;**

**if(!(i%2) && n%2) even++;**

**n/=2;**

**i++;**

**}**

**int ans = 0;**

**if(odd < even){**

**for(int j=1;j<=i;j++){**

**if(j%2 && bin[j-1]==1) continue;**

**else ans+=bin[j-1]\*pow(2,j-1);**

**}**

**}**

**else{**

**for(int j=1;j<=i;j++){**

**if(!(j%2) && bin[j-1]) continue;**

**else ans+=bin[j-1]\*pow(2,j-1);**

**}**

**}**

**cout<<ans;**

**}**

**Wave sort an unsorted array in minimum swaps:**

**#include <iostream>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**int n;**

**cin>>n;**

**int arr[n];**

**for(int i=0;i<n;i++) cin>>arr[i];**

**for(int i=1;i<n;i+=2){**

**if(arr[i-1] > arr[i]) swap(arr[i-1],arr[i]);**

**if(arr[i+1] > arr[i] && i<=n-2) swap(arr[i+1],arr[i]);**

**}**

**for(int i=0;i<n;i++) cout<<arr[i]<<" ";**

**}**

#### **Ticket Number that Yields Maximum Points**

A college is conducting a series of cultural events by inviting celebrities. A group of student volunteers are involved in organising the events. For each event, a student wants to attend, he has to buy a ticket for the event by paying the price mentioned by the organisers. To encourage the participation of students the organisers have announced a type of discount. That is in each ticket for event E1, a random value of point ‘p’ is printed and there will be discount in the price of the ticket for event E2 based on ‘p’. Higher the value of ‘p’, higher the discount will be. When a group of ‘k’ friends come together to buy the tickets, they will be given continuous ‘k’ tickets and they can choose the starting ticket number.

Given the details of the tickets available and number of friends who have come to buy the tickets write a C++ code to determine the first starting ticket number which will yield maximum points for them.

For example, when there are totally nine tickets available and a group of three friends have come to buy tickets and the details of the tickets(ticket number and points) are as follows

10 1

11 2

12 3

13 1

14 4

15 5

16 2

17 3

18 7

They have to choose 16 as starting ticket number to yield maximum points.

**Input Format**

First line contains the total number of tickets available ‘n’ and the number of friends in the group ‘k’ separated by a space

Next ‘n’ lines details on the tickets (ticket number and points printed on the ticket) separated by a space

**Output Format**

Print the starting ticket number that will yield maximum points for the group of friends

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**ios\_base::sync\_with\_stdio(false);**

**cin.tie(NULL);**

**//INPUTS**

**int n,k;**

**cin>>n>>k;**

**vector<int> tk\_no,pts;**

**for(int i=0;i<n;i++){**

**int tk,pt;**

**cin>>tk>>pt;**

**tk\_no.push\_back(tk);**

**pts.push\_back(pt);**

**}**

**//max subarray sum of size k using sliding window which also returns start index**

**int res = 0;**

**for(int i=0;i<k;i++) res+=pts[i];**

**int sum=res,start=0;**

**for(int i=k;i<n;i++){**

**sum+=pts[i]-pts[i-k];**

**if(res < sum){**

**res = sum;**

**start = i-k+1;**

**}**

**}**

**//OUTPUT**

**cout<<tk\_no[start];**

**}**

#### **Wet Will (testament)**

A rich man of a city wrote a will (testament) to distribute his property to his nine children. In addition to lands, houses and jewels, he had also deposited some money in his bank account. The rich man passed away one day and the properties were distributed to his children as per his will. But a part of his will got wet in which one of the digit at some position 'p' of the money that he deposited in his bank account got erased. The rich man’s friend told that the amount (in rupees) deposited by the rich man was such that it can be evenly distributed (in rupees) to his nine children without leaving any remaining rupee(s). For example, if the will contains ₹13456 then the missing digit is 8.

Given the amount written in the will without the missing digit at some position 'p', write a C++ program to find the missing digit so that maximum amount that could have been written on the will.

**Input Format**

First line contains the amount without missing digit at some position 'p', n

**Output Format**

Print the missing digit

**ANS.**

**#include <iostream>**

**#define ll long long**

**using namespace std;**

**int main(){**

**ll n,sum=0; cin>>n;**

**while(n>0){**

**sum+=n%10;**

**n/=10;**

**}**

**for(int i=1;i<=9;i++){**

**if((sum+i)%9 == 0){**

**cout<<i;**

**break;**

**}**

**}**

**}**

#### **Distance from Position in Sliding Puzzle**

All of us might have played this sliding puzzle in our childhood.

****

Now consider a sliding puzzle of dimension nxn. The tiles of all these do not start with 1, and let the starting number of the tiles be ‘s’. Given the shuffled order of the tiles in the puzzle, write a C++ program to find the distance of each tile from its original position. Distance of a tile from its original position is defined as abs (number of rows displaced \*2) + abs (number of columns displaced). For example, if the shuffled tiles are as follows (Empty cell is represented by -1):

41 53 42 44

43 46 48 52

49 47 50 55

45 54 51 -1

and starting tile number is 41 then the distance of the tiles are

41 – 0

53 – 2\*3+1

42 – 1

44 – 0

43 – 2\*1+2

46 – 0

48 – 1

52 – 2\*1

49 – 0

47 – 2\*1+1

50 – 1

55 - 2\*1+1

45 - 2\*2

54 – 0

51 – 2\*1

**Input Format**

First line contains the value of n, dimension of sliding tiles

Next ‘n’ lines contain the value of tiles in each row

**Output Format**

Print the tile number and the distance of the tile from its original position separated by a tab in the order given in the input

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**#include <climits>**

**#include <map>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**int mat[n][n];**

**vector<int> elem;**

**for(int i=0;i<n;i++){**

**for(int j=0;j<n;j++){**

**cin>>mat[i][j];**

**if(mat[i][j] == -1) mat[i][j] = INT\_MAX;**

**elem.push\_back(mat[i][j]);**

**}**

**}**

**// int s; cin>>s;**

**sort(elem.begin(),elem.end());**

**map<int,pair<int,int>> f;**

**int row=0,col=0;**

**for(int i=0;i<n\*n-1;i++){**

**f[elem[i]] = {row,col};**

**col++;**

**if(col > n-1){col=0; row++;}**

**}**

**for(int i=0;i<n;i++){**

**for(int j=0;j<n;j++){**

**if(mat[i][j] == INT\_MAX) continue;**

**cout<<mat[i][j]<<"\t"<<abs(i-f[mat[i][j]].first)\*2+abs(j-f[mat[i][j]].second)<<"\n";**

**}**

**}**

**}**

#### **Reverse Greater**

Here a word S1 is said to be greater than S2, if the number of characters in S1 is greater than S2 or when the number of characters in S1 and S2 are equal and S1 is lexicographically greater than S2.

Given two sentences S1 and S2 with ‘n’ words in them, sentence S1 is said to be reverse greater than sentence S2 if the first word in S1 is greater than the last word (nth word) in S2, second word is lexicographically greater than the last but one (nth word) in S2 .... last word in S1 is greater than the firt word in S2. For example, if the sentences S1 and S2 are ‘thiss iss as sens’ and ‘sen a is this’ then output should be Yes and if the sentences are ‘this is a sen’ and ‘sen a is this’ then the output should be No.

**Hint:**

This problem can be quickly solved with string in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use string, add #include<string>
* We can create object and array of objects for string
* length is a member function of string that gives the number of characters in it
* find\_first\_of(string,pos); – will return the position of the first occurrence of string s where pos is the position of the first character in the string to be considered in the search and return -1 if cannot be found
* find\_last\_of(string,pos); – will return the position of the last occurrence of string s where pos is the position of the last character in the string to be considered in the search and return -1 if cannot be found
* substr(i,n) – will return a substring from position ‘i’ and of length ‘n’

**Sample Code using String in STL**

#include<iostream>

using namespace std;

#include<string>

int main()

{

string s1,s2;

int l,pos;

getline(cin,s1);

//get length of s1

l=s1.length();

//substring of length 3 from position 2

s2 = s1.substr(2,3);

//searched for first occurrence of string s2 in s1 from position 3

l = s1.find\_first\_of(s2,3);

**}**

**Input Format**

First line contains first sentence S1

Next line contains second sentence S2

**Output Format**

Print Yes or No

**ANS.**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main(){**

**string s1,s2;**

**getline(cin,s1);**

**getline(cin,s2);**

**vector<string> words1,words2;**

**string word="";**

**for(int i=0;i<=s1.length();i++){**

**if(s1[i] == ' ' || s1[i] == '\0'){**

**words1.push\_back(word);**

**word="";**

**}**

**else word+=s1[i];**

**}**

**word = "";**

**for(int i=0;i<=s2.length();i++){**

**if(s2[i] == ' ' || s2[i] == '\0'){**

**words2.push\_back(word);**

**word="";**

**}**

**else word+=s2[i];**

**}**

**int m = words1.size();**

**string ans = "";**

**for(int i=0;i<m;i++){**

**if(words1[i].length() > words2[m-i-1].length()) ans+='1';**

**else if(words1[i].length() < words2[m-i-1].length()) ans+='2';**

**else{**

**if(words1[i] < words2[m-i-1]) ans+='2';**

**else if(words1[i] > words2[m-i-1]) ans+='1';**

**else ans+='0';**

**}**

**}**

**for(int i=0;i<ans.length();i++){**

**if(ans[i] != '1') {**

**cout<<"No";**

**return 0;**

**}**

**}**

**cout<<"Yes";**

**}**

#### **Colored Coin Game**

**There was a mistake in the test case Apologies for that. Now it is corrected**

Colored coin game is a two player game and coins of three colors red (r), green(g) and blue (b) are used. Initially ‘n’ coins of some color are given to both the players. A die was thrown and as per the number that falls the follwing operations will be performed. When there are initially seven coins with the players and they are r, g, b, r, g, b, r the result on throwing the dice with the number is shown.

1- Reassign - r, g, b, r, g, b, r – Assignment should be taken care so that minimum of the number of the elements in the vectors are copied

2 - Insert coin – Insert red coin at position 2 – r, r, g, b, r, g, b, r

3 – Insert four coins – Insert four red coins from position 3 - r, g, r, r, r, r, b, r, g, b, r

4 – Remove one coin at position 4 - r g b g b r

5 – Remove two coins from position 5 - r g b r r

6 – Remove all coins

Given the initial coins in the hands of the players and number got in ‘n’ dice throw for both the player, write a C++ program to print the coins in the hands of the players.

**Hint**

This problem can be quickly solved with vector in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use vector, add #include<vector>
* We can create object and array of objects for vector
* size() - Number of elements in a vector can be found
* [] - can be used to random access elements of a vector
* begin() - will give reference to the first element in the vector which can be stored in an iterator
* end() - will give reference to one position next to last element in the vector which can be stored in an iterator
* iterator for a integer vector can be declared as vector<int>::iterator it
* +, -, ++, --, <, >, ==, != - operators that can be used with iterators
* insert(pos, e) – Inserts the element ‘e’ at the given iterator position pos
* erase(pos) – Removes the element at the given iterator position pos
* erase(pos1,pos2) – Removes the element2 at from given iterator position pos1 to iterator position p2
* assign(it1,it2) – It assigns new value to the vector elements by replacing old ones from iterator 1 to iterator 2 – similar to copying elements of one vector into another

**Sample code**

#include<iostream>

using namespace std;

#include<vector>

int main()

{

vector<int> v1,v2;

int ele;

v1.push\_back(11);

v1.push\_back(12);

vector<int>::iterator it =v1.begin();

//to delete data at 5th position

v1.erase(it+4);

//copy all content of v1 into v2

v2.assign(v1.begin(),v1.begin()+v1.size());

//to delete data from 5th position to 8th position

v1.erase(it+4, it+7);

//to insert data at 6th position

v1.insert(it+5, ele);

for(int i=0;i<v.size();i++)

{

cout<<v[i]<<endl;

}

//to check if elements of two vectors are same

if(v1==v)

cout<<"Same";

}

**Input Format**

First line contains the number of coins initially given, n

Next line contains the color of the coins in initially given to both players separated by a space

Next line contains the number of dice throws, n1

Next line contains n1 numbers got by player1

Next line contains n1 numbers got by player2

**Output Format**

First line print the coins in hand of player1 separated by a space

Next line print the coins in hand of player2 separated by a space

If no coins is left then print “No coins with player1" or “No coins with player2"

**Note:**There is a space at the end of both the lines

**ANS.**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**void func(int d,vector<char> p,vector<char> e,int no){**

**for(int i=0;i<d;i++){**

**int ch; cin>>ch;**

**switch(ch){**

**case 1:{**

**p = e; //VECTORS CAN BE ASSIGNED**

**break;**

**}**

**case 2:{**

**p.insert(p.begin()+1,'r');**

**break;**

**}**

**case 3:{**

**for(int i=0;i<4;i++) p.insert(p.begin()+2,'r');**

**break;**

**}**

**case 4:{**

**if(p.size()<4) goto end;**

**p.erase(p.begin()+3);**

**break;**

**}**

**case 5:{**

**if(p.size()<5) goto end;**

**p.erase(p.begin()+4);**

**if(p.size()<5) goto end;**

**p.erase(p.begin()+4);**

**break;**

**}**

**case 6:{**

**p.clear();**

**if(i == d-1) goto end;**

**break;**

**}**

**default: break;**

**}**

**}**

**for(auto x: p) cout<<x<<" ";**

**cout<<"\n";**

**return;**

**//USE GOTO LABEL IN CASES YOU DON’T WANT TO PUT EXIT BUT NEED SIMILAR FUNCTIONALITY**

**end: cout<<"No coins with player"<<no<<"\n";**

**}**

**int main(){**

**int n; cin>>n;**

**vector<char> e(n),p1(n),p2(n);**

**for(int i=0;i<n;i++){**

**cin>>e[i];**

**p1[i]=e[i];**

**p2[i]=e[i];**

**}**

**int d; cin>>d;**

**func(d,p1,e,1);**

**func(d,p2,e,2);**

**}**

#### **Element Names and Symbols**

A child is studying sixth class and he has got a list of elements and the corresponding elements in his science book. It will be helpful to make him to learn if a program could give the name of the element if symbol is given vice versa. Given the elements and their symbols in the child’s book, write a C++ program to print the name or symbol as required. For example, if there are five elements with name and symbol as (zinc, zn), (sulphur, s), (chlorine, cl), (sodium, na) and (tin, sb) and symbol is given as sb then print tin.

**Hint**

This problem can be quickly solved with map in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

* To use map, add #include<map>
* We have to give the data type of key and value during object creation. For example, map<int, int> m will create a map that can store an integer key and an integer value.
* Value can be inserted into the map m as m[key] = value
* To check if a key is present in a map, use count function. It returns 0 – if key is not present and 1 – otherwise
* To take the value of an existing key in the map use value = m[key]

**Input Format**

First line contains the number of elements in his book, n

Next ‘n’ lines contain name of element followed by symbol separated by a space

Next line choice of query, 1 – symbol of element will be given 2- name of element will be given

**Output Format**

Print Symbol or Element name if present otherwise print “Not found”

**ANS.**

**#include <iostream>**

**#include <map>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**map<string,string> table;**

**while(n--){**

**string ele,sym;**

**cin>>ele>>sym;**

**table[sym] = ele;**

**}**

**int ch; string in;**

**cin>>ch>>in;**

**for(auto x: table){**

**if(x.first == in) {cout<<x.second; return 0;}**

**else if(x.second == in) {cout<<x.first; return 0;}**

**}**

**cout<<"Not found";**

**}**

#### **Maximum Common Difference Element in the Right**

Given a collection of ‘n’ integers, write a C++ program to find the difference between the elements to the right that has occurred maximum number of times. Print the elements with the maximum common difference in ascending order. For example, given six numbers, 28, 32, 36, 38, 34 and 30 then the numbers and the difference with the numbers in their right are

28 - 4 8 10 6 2

32 - 4 6 2 2

36 - 2 2 6

38 - 4 8

34 - 4

common difference that occurs maximum number of times in distinct elements is 4 and the elements are 28 32 34 38 in ascending order.

**Hint:**

This problem can be easily solved with containers such as list, vector in STL and sort, find algorithms in STL

**Input Format**

First line contains the number of elements, n

Next line contains the ‘n’ elements separated by a space

**Output Format**

Print the common difference that has occurred maximum number of times, m

In next line, print the elements that has got the difference ‘m’ with the elements in the right, separated by a space

**ANS.**

**PYTHON (WORKS IN VPROPEL):**

**n = int(input())**

**arr = list(map(int,input().split()))**

**diff = {}**

**for i in range(n-1):**

**vis = set()**

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

**d = abs(arr[i]-arr[j])**

**if d not in diff.keys(): diff[d] = [arr[i]]; vis.add(d)**

**elif d not in vis: diff[d].append(arr[i]); vis.add(d)**

**max\_arr,ans = [],-1**

**for i in diff.keys():**

**if len(diff[i]) > len(max\_arr): max\_arr=diff[i]; ans=i**

**print(ans); print(\*sorted(max\_arr))**

**C++ CODE (WORKS BUT UNEXPECTED ERROR IN VPROPEL):**

**#include <iostream>**

**#include <vector>**

**#include <set>**

**#include <algorithm>**

**#include <unordered\_map>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**vector<int> arr(n);**

**unordered\_map<int,vector<int>> diff;**

**set<int> vis;**

**for(int i=0;i<n;i++) cin>>arr[i];**

**for(int i=0;i<n-1;i++){**

**vis.clear();**

**for(int j=i+1;j<n;j++){**

**int d = abs(arr[i]-arr[j]);**

**if(diff.find(d) == diff.end()) {diff[d] = {arr[i]}; vis.insert(d);}**

**else if(vis.find(d) == vis.end()) {diff[d].push\_back(arr[i]); vis.insert(d);}**

**}**

**}**

**vector<int> max\_arr; int ans = -1;**

**for(auto x: diff){**

**if((x.second).size() > max\_arr.size()) {max\_arr = x.second; ans = x.first;}**

**}**

**sort(max\_arr.begin(),max\_arr.end());**

**cout<<ans<<"\n";**

**for(auto x: max\_arr) cout<<x<<" ";**

**}**

#### **Swap Coins**

Given the ‘m’ coins held by ‘n’ children, write a C++ program to perform ‘k’ swap coins operations. During swap coins operation between two children, all the coins held by one child is swapped with all coins of the other child. After swap operations, print the coins held the children in the order as given in input. For example, when there are five children with five coins as follows:

Name1 5 10 12 45 67

Name2 7 9 11 13 49

Name3 90 21 34 56 67

Name4 22 35 68 91 29

Name5 37 78 93 44 55

After three swaps between the children mentioned below:

Name3 Name5

Name2 Name1

Name5 Name4

Coins with the children left over are:

Name1 7 9 11 13 49

Name2 5 10 12 45 67

Name3 37 78 93 44 55

Name4 90 21 34 56 67

Name5 22 35 68 91 29

**Hint:**

This problem can be quickly solved using map and vector in STL. Vector has a function swap which will swap elements of two vectors v1 and v2.

For example, v1.swap(v2) – will swap all elements of v1 and v2

**Input Format**

First line contains the number of children, n

Next ‘n’ lines contain the name of the child and the coins the coins held by the child

Next line contains the value of k

Next ‘k’ lines contain the name of the two children whose coins are to be swapped

**Output Format**

Print ‘n’ lines

In ith line, print name of the ith child and the coins in the hand of the ith child

**Note:**A space is there at the end of each line

**ANS.**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int num\_extract(string f){**

**if(f.length() == 6)**

**return (f[4]-'0')\*10+(f[5]-'0')-1;**

**return f[4]-'0'-1;**

**}**

**int main(){**

**int n,m; cin>>n>>m;**

**vector<vector<string>> info;**

**for(int i=0;i<n;i++){**

**vector<string> ele(m+1);**

**for(int j=0;j<m+1;j++) cin>>ele[j];**

**info.push\_back(ele);**

**}**

**int k; cin>>k;**

**for(int i=0;i<k;i++){**

**string f,s; cin>>f>>s;**

**int i1 = num\_extract(f),i2 = num\_extract(s);**

**info[i1].swap(info[i2]);**

**swap(info[i1][0],info[i2][0]);**

**}**

**for(auto x: info){**

**for(auto y: x) cout<<y<<" ";**

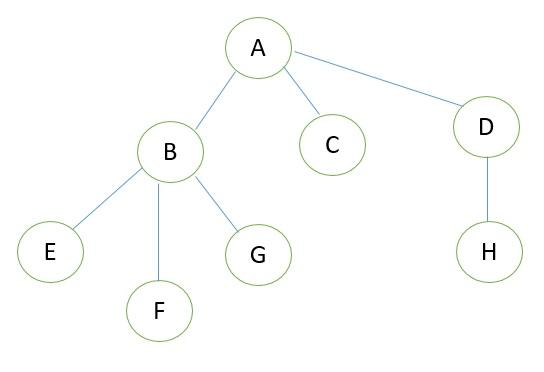
**cout<<"\n";**

**}**

**}**

#### **Dependency Tree**

A dependency tree organizes structures so there are explicit relationships between entities. In a tree structure, each node can have only one parent but a node may have many children. A dependecy tree can drawn between jobs, softwares or items. For example, consider the dependency tree figure given below:

****

A is referred as parent of B, C and D, B is parent of E, F and G and E, F and G are children of B and so on. Given the details of all parent and child relationship in a dependency tree and a name ‘m’, write a C++ code to find all dependencies of ‘m’. The above tree structure contain seven parent child relation as follows:

A B

A C

B E

B F

B G

D H

For example, in the above tree, if ‘F’ is given then the dependencies are B and A.

**Input Format**

Number of parent child realtion in the tree, n

Next ‘n’ lines contain the name of the parent and the child nodes

Next line contains the name of the node whose dependency is to be found ‘m’

**Output Format**

Print the dependency node names from direct parent of the node ‘m’ in a line separated by a space

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <map>**

**#include <algorithm>**

**using namespace std;**

**void func(map<char,vector<char>> &tree,char m){**

**int cnt = 0; char next;**

**for(auto x: tree){**

**if(find(x.second.begin(),x.second.end(),m) != x.second.end()){**

**cout<<x.first<<" ";**

**next = x.first;**

**cnt = 1;**

**break;**

**}**

**}**

**if(cnt == 0) return;**

**else func(tree,next);**

**}**

**int main(){**

**int n; cin>>n;**

**map<char,vector<char>> tree;**

**for(int i=0;i<n;i++){**

**char parent,child;**

**cin>>parent>>child;**

**if(tree.find(parent) == tree.end()) tree[parent] = {child};**

**else tree[parent].push\_back(child);**

**}**

**char m; cin>>m;**

**func(tree,m);**

**}**

**Distribute T-Shirts in a Marathon**

A marathon is being conducted by a NGO in Chennai. T-Shirts with the theme of the marathon and a number is given to the participants. The participants are given the T-Shirts in the order of their arrival and new T-Shirts are also continuously brought-in to the venue. T-Shirts are arranged one above the other vertically and they are distributed from top to bottom. Three operations are possible partcipant-entry, t-shirt-arrival and distribute t-shirt.

**Participant-entry** - Some ‘p’ number of participants are arrived and they are made to wait for t-shirt distribution.

**T-shirt arrival** – Some ‘m’ number of t-shirts are brought to the venue and arranged on a table. Always t-shirts arrived recently will be on the top

**Distribute-t-shirt** – Some ‘k’ number of t-shirts are distributed to ‘k’ participants, print name of the participant and t-shirt number

Given a sequence of operations performed, write a C++ code to carry out and print the details of t-shirts distributed to partcipants during distribute-t-shirt operation.

**Illustration**

If the input is

7

1

2

Name1 Name2

2

3 6 9 4

1

3 Name3 Name5 Name4

2

7 8 5 1 45 56 78 12

1

2 Name12 Name6

1

1 Name8

3

4

then the desired output is

Name1 12

Name2 78

Name3 56

Name5 45

**Input Format**

First line contains the number of operations, n

Details for ‘n’ operations are given. Code for operations, 1- Participant-entry, 2 - T-shirt arrival and 3 – Distribute-t-shirt

For option 1, next line will contain the number of participants entered currently, n, followed by ‘n’ names

For option 2, next line will contain the number of t-shirts arrived, m, followed by ‘n’ numbers written on the t-shirts

For option 3, next line contains the value of k

**Output Format**

For option 3, print the name and the t-shirt number given for the participant separated by a space

**ANS.**

**#include <iostream>**

**#include <stack>**

**#include <queue>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**stack<int> shirts;**

**queue<string> participants;**

**for(int i=0;i<n;i++){**

**int option; cin>>option;**

**switch(option){**

**case 1:{**

**int nop; cin>>nop;**

**for(int i=0;i<nop;i++){**

**string p; cin>>p;**

**participants.push(p);**

**}**

**break;**

**}**

**case 2:{**

**int nos; cin>>nos;**

**for(int i=0;i<nos;i++){**

**int sn; cin>>sn;**

**shirts.push(sn);**

**}**

**break;**

**}**

**case 3:{**

**int k; cin>>k;**

**while(k-- && !participants.empty() && !shirts.empty()){**

**cout<<participants.front()<<" "<<shirts.top()<<"\n";**

**participants.pop(); shirts.pop();**

**}**

**break;**

**}**

**default: {cout<<"invalid input"; break;}**

**}**

**}**

**}**

#### **ith Digit Middle Number**

In a collection of ‘n’ numbers with same number of digits, a number ‘m’ in postion ‘p’ is said to be an ith digit middle number if the ith digit of m is greater than or equal to the ith digit of all numbers in position < p and ith digit of m is lesser than ith digit of all numbers in position > p. First number in the collection is said to ith digit middle number if the ith digit is lesser than ith digit of all numbers to its right and the last number in the collection is said to be ith digit middle number if its ith digit is greater than or equal to ith digit of all numbers to its left. First digit of the number is the left most digit.

Given a collection of numbers, C1 and a digit position i, write a C program to print all ith digit middle numbers in C1.

For example, given a collection of five, four digits numbers 7632, 9645, 1834, 2778, 9956 the ith digit middle number when i = 2 are 9645 and 9956.

**Input Format**

First line contains the number of elements in collection C, n

Next ‘n’ lines contain the numbers

Next line contains the position ‘i’

**Output Format**

Print all the ith digit middle numbers

Print one number in one line in same order as given in input

**ANS.**

**#include <iostream>**

**using namespace std;**

**bool check(string arr[],int n,int ind,int pos){**

**for(int i=0;i<ind;i++){**

**if(arr[i][pos-1] > arr[ind][pos-1]) return false;**

**}**

**for(int i=ind+1;i<n;i++){**

**if(arr[i][pos-1] <= arr[ind][pos-1]) return false;**

**}**

**return true;**

**}**

**int main(){**

**int n; cin>>n;**

**string arr[n];**

**for(int i=0;i<n;i++) cin>>arr[i];**

**int pos; cin>>pos;**

**for(int i=0;i<n;i++){**

**if(check(arr,n,i,pos)) cout<<arr[i]<<"\n";**

**}**

**}**

#### **Cut the Wood**

A carpenter has got a wooden block of length ‘l’ and he has got four automated cutters that can cut the wood into smaller sizes of some standard sizes so that the wood could be stored in his godown. During the cutting process one portion of wooden piece shall be wasted. He wants to cut them into either minimum number of pieces or maximum number of pieces. For example, given the length of the wood as 253 units, length of the standard pieces that can be cut by the wooden cutter 4, 17, 7 and 2 the minimum number of pieces that can be got is 16 with a wastage of 1 unit. The maximum number of pieces that can be made with the cutters is 126 with a wastage of 1 unit. Given the length of the wooden piece l, and the four different standard sizes, write a C program to find the minimum and maximum number of pieces that can be formed and the wastage in each case.

**Input Format**

First line contains the first standard size, a

Next line contains the first standard size, b

Next line contains the first standard size, c

Next line contains the first standard size, d

Next line contains the length of the wood, l

**Output Format**

In the frist line, print the minimum number of wooden pieces that can be formed and wastage separated by a tab

In the Next line, print the maximum number of wooden pieces that can be formed and wastage separated by a tab

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**int a,b,c,d,l; cin>>a>>b>>c>>d>>l;**

**vector<int> sizes = {a,b,c,d};**

**sort(sizes.begin(),sizes.end());**

**//for min number of pieces with min wastage take the max of a,b,c,d smaller**

**//than l at the moment then**

**//divide l with it and for the remainder repeat the process**

**//until you cant because l is smaller than min(a,b,c,d)**

**int min = 0,lc = l;**

**for(int i=3;i>=0;i--){**

**if(sizes[i] <= lc) {**

**min+=lc/sizes[i];**

**lc=lc%sizes[i];**

**}**

**}**

**cout<<min<<"\t"<<lc<<"\n";**

**//similar but here you choose the min of a,b,c,d at the moment less than l**

**int max = 0; lc = l;**

**for(int i=0;i<4;i++){**

**max+=lc/sizes[i]; lc=lc%sizes[i];**

**}**

**cout<<max<<"\t"<<lc;**

**}**

#### **Speaking Birds**

Consider an imaginary world, where all organisms can communicate with each other. On a particular day, ‘**n’**number of birds were speaking to Venkat one after the another. First bird always says a '1', and other birds describes the speech of the previous bird. Given ‘**n’**, help Venkat to predict the length of the speech of the nth bird.

For example, if 'n' is 5 then the first bird speaks ‘1’, second bird speaks ‘11’, third bird speaks ‘21’, fourth bird speaks ‘1211’ and the fifth bird speaks ‘111221’.

**Sample input**

5

**Sample Output**

6

**Input Format**

First line contains the value of ‘n’

**Output Format**

Print the length of the speech by the nth bird

Constraints : 5<=n<=35

**ANS.**

**#include <iostream>**

**using namespace std;**

**void func(string &start,int n){**

**if(n==1) return;**

**string next = "";**

**int f=1,m=start.length(); char key=start[0];**

**for(int i=1;i<=m;i++){**

**if(start[i] == key) {f++; continue;}**

**else if(i == m || start[i]!=key){**

**next+=(char)'0'+f;**

**next+=key;**

**if(i < m){**

**key = start[i];**

**f=1;**

**}**

**}**

**}**

**start = next;**

**func(start,n-1);**

**}**

**int main(){**

**int n; cin>>n;**

**string start = "1";**

**func(start,n);**

**cout<<start.length();**

**}**

#### **New House**

Siddharth wants to build a new house. He wants to list out all the possible expenses in order apply for loan. He approaches a building contractor who first decides the total number of components, i.e. furniture, electronics etc.Then the contractor grouped the components into various section that’ll roughly cost the same. Siddharth wantsto identify the cost associated with most expensive component(s), help siddardath by writing a code to find it out.

For example if there are 10 components in total and the contractor divides them into 4 sections as follows:

First section is from 2nd component to 6th component in which cost of each component is roughly ₹8000

Second section is from 3rd component to 5th component in which cost of each component is roughly ₹7000

The 3rd section is from 1st component to 8th component in which cost of each component is roughly ₹1000

The 4th section is from 5th component to 9th component in which cost of each component is roughly ₹15000

The cost associated with the most expensive component in this is ₹31000.

**NOTE**

This problem is expected to be solved without using nested loops in any section of the code

**INPUT FORMAT**

The first line contains two space-separated integers **n** and **m** , the number of components and the number of sections.  
Each of the next **m** lines contains three space-separated integers the first component, the last component and price for that section.

**OUTPUT FORMAT**

The price associated with the most expensive component

**ANS.**

**PYTHON (OPTIMISED):**

**n,m = map(int,input().split())**

**c = [0 for i in range(n+2)]**

**for i in range(m):**

**x,y,z = map(int,input().split())**

**c[x]+=z; c[y+1]+=-z**

**ans,sum = -1,0**

**for i in c:**

**sum+=i**

**ans = max(ans,sum)**

**print(ans)**

**C++ CODE (BRUTE FORCE):**

**#include <iostream>**

**#include <vector>**

**#include <climits>**

**#define int long long**

**using namespace std;**

**int32\_t main(){**

**int n,m; cin>>n>>m;**

**vector<int> f(n+1,0);**

**for(int i=0;i<m;i++){**

**int r1,r2,price;**

**cin>>r1>>r2>>price;**

**for(int j=r1;j<=r2;j++) f[j]+=price;**

**}**

**int ans = INT\_MIN;**

**for(auto x: f) ans = max(ans,x);**

**cout<<ans;**

**}**

#### **Help the teacher**

Tarun is a English teacher.On a particular day he asks his students to write a word in a paper.He says that all characters of the string appear the same number of times.He also says that the word must atleast have a length of 25 words.After every students write he collects the papers and starts correcting.If in the wordall character appear the same number of times he gives them full marks.Tarun being a lenient teacher gives full marks if he removes 1 character from the written word and then all characters of the string appear the same number of times. But Tarun is very lazy to check all the answers and he requires your help. Given a word print YES if he will get full mark or NO otherwise.

For example if s=abc the string is valid as frequencies of a,b,c are all 1.So is abcc as if u remove one c then the frequencies of each character will be equal.However abccc is not a valid string as removing any character does not result with string where frequencies of all characters are same.

**ANS.**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main(){**

**string s; cin>>s;**

**vector<int> f(27,0);**

**for(char c: s){**

**if(!f[(int)c-'a'+1]) f[(int)c-'a'+1]++;**

**else f[(int)c-'a'+1]--;**

**}**

**int sum = 0;**

**for(auto x: f) sum+=x;**

**if(sum != 0 && sum != 1) cout<<"NO";**

**else cout<<"YES";**

**}**

**PYTHON:**

**s = input()**

**f = [0 for i in range(27)]**

**for i in s:**

**if f[ord(i)-96] == 0: f[ord(i)-96]+=1**

**else: f[ord(i)-96]-=1**

**chk = sum(f)**

**if(chk != 0 and chk != 1): print("NO")**

**else: print("YES")**

**JAVA:**

**import java.util.\*;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**String s = sc.nextLine();**

**int f[] = new int[27];**

**for(int i=0;i<s.length();i++){**

**if(f[(int)s.charAt(i)-'a'+1] == 0) f[(int)s.charAt(i)-'a'+1]++;**

**else f[(int)s.charAt(i)-'a'+1]--;**

**}**

**int sum = 0;**

**for(int c: f) sum+=c;**

**if(sum == 1 || sum == 0) System.out.println("YES");**

**else System.out.println("NO");**

**}**

**}**

#### **Building Blocks**

Venkat is playing building blocks. He has blocks of various lengths. He arranges the blocks horizontally in some random order. He observes that there are some spaces created in between the blocks. He decides to keep a ball in every free space generated. Given the heights of various block help Venkat to find the number of balls he can fit in the space.

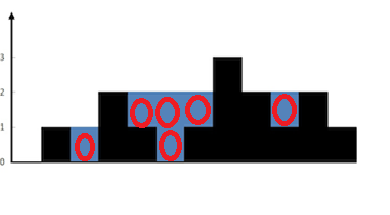
Assume that the size of a free space is exactly the same as the size of the circle.

**Example:-**

Let the lengths of various blocks be

[0,1,0,2,1,0,1,3,2,1,2,1]

The above can be visualized as follows. Every square generated is equivalent to a free space. So there are six free spaces. So the expected answer is six

****

**ANS.**

**n = int(input())**

**b = [int(input()) for i in range(n)]**

**left,right = [b[0]],[b[-1]]**

**#calculating left side maximums for each bar**

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

**if(left[-1] < b[i]): left.append(b[i])**

**else: left.append(left[-1])**

**#calculating right side maximums for each bar**

**for i in range(n-2,-1,-1):**

**if(right[0] < b[i]): right.insert(0,b[i])**

**else: right.insert(0,right[0])**

**#water that can be stored at each bar is minimum of one of the boundaries and height of bar**

**water = 0**

**for i in range(1,n-1):**

**var = min(left[i-1],right[i+1])**

**if(var > b[i]): water+=var-b[i]**

**print(water)**

**TWO POINTER C++ SOLUTION:**

**#include <iostream>**

**#include <vector>**

**#define long long int**

**using namespace std;**

**int trap(vector<int>& height,int n){**

**ios\_base::sync\_with\_stdio(false);**

**cin.tie(NULL); cout.tie(NULL);**

**int lmax=0,rmax=0,left=0,right=n-1,water=0;**

**while(left <= right){**

**if(rmax <= lmax){**

**water+=max(0,rmax-height[right]);**

**rmax = max(rmax,height[right]);**

**right--;**

**}**

**else{**

**water+=max(0,lmax-height[left]);**

**lmax = max(lmax,height[left]);**

**left++;**

**}**

**}**

**return water;**

**}**

**int32\_t main(){**

**int n; cin>>n;**

**vector<int> height(n);**

**for(int i=0;i<n;i++) cin>>height[i];**

**cout<<trap(height,n);**

**}**

#### **Collect the stamps**

Siddharth has a hobby of collecting stamps. He has k bags to put the stamps. Each bag can have only one of type stamp. You will be given with an array where each element represents the type of that stamp. Help Siddharth find the maximum number of stamps he can collect without emptying any one of his bag

Example:-

Let the stamp type array be [3,3,3,1,2,1,2,1,2,1,2,2,4] and he has 2 bags to put the stamps

Here 1,2,3,4 denotes the type of stamp, so there are 4 types of stamp

He can collect the stamps [1,2,1,2,1,2,1,2,2] without emptying any of his bag as it has two types of stamps

So the maximum number of stamps he can collect without emptying any basket is 9

**Input format:-**

Number of stamps

Next n lines-type of each stamp

number of  bags

**Output format:-**

The maximum number of stamps he can collect without emptying any basket

**ANS.**

**n = int(input())**

**s = [int(input()) for i in range(n)]**

**b = int(input())**

**ans = 0**

**for i in range(n-1):**

**p = set(); p.add(s[i]); cnt = 1**

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

**p.add(s[j]); cnt+=1**

**if(len(p) > b): ans = max(ans,cnt-1); break**

**if(j == n-1):**

**if(len(p) > b): ans = max(ans,cnt-1)**

**else: ans = max(ans,cnt)**

**break**

**if ans > n//2: break**

**print(ans)**

**JAVA CODE:**

**import java.util.\*;**

**class Main{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**int n = sc.nextInt();**

**int arr[] = new int[n];**

**for(int i=0;i<n;i++) arr[i] = sc.nextInt();**

**int b = sc.nextInt();**

**int ans = -1;**

**for(int i=0;i<n-1;i++){**

**HashSet <Integer> hs = new HashSet <Integer> ();**

**int cnt = 1;**

**hs.add(arr[i]);**

**for(int j=i+1;j<n;j++){**

**hs.add(arr[j]); cnt++;**

**if(hs.size() > b){**

**ans = Math.max(ans,cnt-1);**

**break;**

**}**

**if(j == n-1){**

**if(hs.size() > b) ans = Math.max(ans,cnt-1);**

**else ans = Math.max(ans,cnt);**

**break;**

**}**

**}**

**if(ans > n/2) break;**

**}**

**System.out.println(ans);**

**}**

**}**

#### **No Change**

Vikram’s mom gave him a certain amount of money and asked him to get change for the money in coins.

She also told the denominations of coin she wanted and also told the total number of coins must be minimum.

If that amount of money cannot be made up by any combination of the coins, print -1.

**Example1:-**

Let the Amount – 11

Let the Denominations be = [1, 2, 5]

**Output:- 3**

Explanation:-  11 = 5+5+1 (This is the minimum possible combination)

Example2:-

Let the amount:- 3

Let the denominations be:- [2]

Output:-  -1

Explanation:- 3 cannot be made up by any combination of the coins of denomination 2

**Input format:-**

Total Amount

Number of denomination-n

Next n lines-denomination

**Output format:-**

Minimum number of coins.

**Constraints:-**

1<=denominations[i]<=30

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <climits>**

**using namespace std;**

**int coinChange(vector<int> &coins, int t) {**

**vector<int> dp(t+1,INT\_MAX-1);**

**dp[0]=0;**

**for(int i=0;i<t+1;i++){**

**for(int j=0;j<coins.size();j++){**

**if(i>=coins[j]) dp[i]=min(dp[i],1+dp[i-coins[j]]);**

**}**

**}**

**if(dp[t]<INT\_MAX-1) return dp[t];**

**return -1;**

**}**

**int main()**

**{**

**int sum,n; cin>>sum>>n;**

**vector<int> coins(n);**

**for(int i=0;i<n;i++) cin>>coins[i];**

**cout<<coinChange(coins,sum);**

**return 0;**

**}**

#### **Point to Point**

Shyam is standing in Point A. He has to reach Point B. There are m number of divisions

between Point A and point B.

These division may or may not have a stone. One can jump only on the stone.

Given a list of stones positions (in division) in sorted ascending order,

determine whether Shyam can reach Point B.

Initially Shyam is at the first stone and assume the first jump is one division

If Shyam’s last jump is k divisions the he can either     k-1 or k or k+1 divisions in the next jump.

He can jump only in forward direction.

**Example1:-**

Let The position of stones be [0,1,3,5,6,8,12,17]

There are a total of 8 stones.

The first stone at the 0th division, second stone at the 1st division, third stone at the 3rd division, and so on...

The last stone at the 17th division.

Output:-Yes

**Explanation:-**

Shyam can jump to the last stone by jumping  1 division to the 2nd stone, then 2 divisions to the 3rd stone, then  2 divisions to the 4th stone, then 3 divisions to the 6th stone, 4 divisions to the 7th stone, and 5 divisions to the 8th stone.

**Example2:-**

Let the positions of stones be [0,1,2,3,4,8,9,11]

Output:-No

**Explanation:-**

There is no way to jump to the last stone as

the gap between the 5th and 6th stone is too large.

**Input format:-**

Number of stones

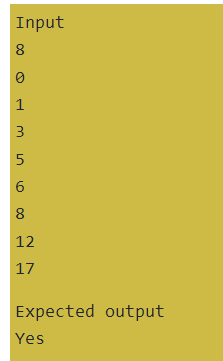
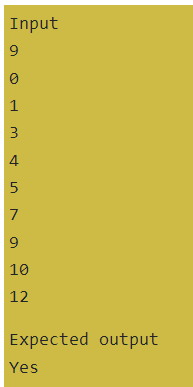
Next n lines –positions of n stones

**Output format:-**

Yes or No

**Note:-** The first stone is always in the 0th division

**ANS.**

**  **

#### **Help the doctor**

Sabarish, A doctor forms a grid where each cell represents the condition of a patient.

The cell can take any of the three values as follows:

              1-The person is virus free

              2-The person is infected with the virus

               3-Empty cell

It takes one day for the virus to spread from one person to another.

Every day any virus free person who is adjacent (4-directionally) to the infected person catches the disease.

Given the initial grid help the doctor to find the minimum number of days that must elapse until

everyone has been infected. If this impossible return -1.

**Example:-**

Suppose the input grid is 2 1 1

                                            1 1 3

                                            3 1 1

After day 1 the grid will be    2 2 1

                                                  2 1 3

                                                  3 1 1

After day 2 the grid  will be  2 2 2

                                                 2 2 3

                                                 3 1 1

After day 3 the grid will be 2 2 2

                                               2 2 3

                                               3 2 1

After day 4 the grid will be 2 2 2

                                               2 2 3

                                               3 2 2

So, the minimum number of days is 4

**Input format:-**

Numbers of row in the grid

Numbers of column in the grid

Contents of the grid(next row\*column lines)

**Output format:-**

Minimum number of days required

**ANS.**

**direction = [(1, 0), (0, -1), (-1, 0), (0, 1)]**

**def maximumTime(arr):**

**n = len(arr)**

**m = len(arr[0])**

**visited = [[False for j in range(m)] for i in range(n)]**

**q = []**

**uninfected\_count = 0**

**time = 0**

**for i in range(n):**

**for j in range(m):**

**if arr[i][j] == 2:**

**q.append(((i, j), 0))**

**visited[i][j] = True**

**if arr[i][j] == 1:**

**uninfected\_count += 1**

**while q:**

**current = q.pop(0)**

**time = current[1]**

**for it in direction:**

**i = current[0][0] + it[0]**

**j = current[0][1] + it[1]**

**if i < 0 or j < 0 or i >= n or j >= m or arr[i][j] != 1 or visited[i][j]:**

**continue**

**q.append(((i, j), time + 1))**

**visited[i][j] = True**

**uninfected\_count -= 1**

**if uninfected\_count != 0:**

**return -1**

**return time**

**r = int(input())**

**c = int(input())**

**m = []**

**for i in range(r):**

**m.append([])**

**for j in range(c):**

**m[i].append(int(input()))**

**print(maximumTime(m))**

#### **Eat chocolates**

Ramesh wants to buy some chocolates. But there are some constraints that she needs to follow. She can only buy “m” chocolates and can buy the chocolates of the same brand only up to a certain limit. Given the cost and brand of the chocolates find the maximum cost at which she can buy the chocolates

**Example1:-**

Cost of chocolates:- [5,4,3,2,1]

Cost[i] represents the cost of ith chocolate

Brand of the chocolates :-[1,1,2,2,3]

Here brand[i] represents the brand of its chocolates 1,2,3 etc. represents brand

No.of Chocolates to be bought-3

Brand Limit -1

In this case she can select only one chocolate per brand as the brand limit is one

So the  combination which leads to maximum cost is 5+3+1=9

Each of the brand has been used lesser than or equal to the brand limit

**Example2:-**

Cost of chocolates:-[5,4,3,2,1]

Brand of chocolates:-[1,3,3,3,2]

No of chocolates to be bought-3

Brand limit -2

In this she can choose upto two chocolates of the same brand

So in this case the combination 5+4+3 leads to maximum cost

**Constraints:-**

1<=brand[i]<=5(There are totally 5  brands)

**Input format:-**

n-Total number of chocolates in shop

Next n lines –Cost of chocolates

Next n lines –Brand of chocolates

Number of chocolates to be bought(m)

Brand Limit

**Output:-**

Maximum cost

**ANS.**

**n = int(input())**

**cost = [int(input()) for i in range(n)]**

**brand = [int(input()) for i in range(n)]**

**m = int(input())**

**bl = int(input())**

**info = dict(zip(cost,brand))**

**cost.sort(reverse = True)**

**brlimt = {}**

**for i in brand:**

**c = brand.count(i)**

**if(c < bl): brlimt[i] = c**

**else: brlimt[i] = bl**

**ans = 0**

**try:**

**while(m > 0):**

**if(brlimt[info[cost[0]]] > 0):**

**ans+=cost[0]**

**brlimt[info[cost[0]]]-=1**

**cost.pop(0)**

**m-=1**

**else: cost.pop(0)**

**except: print(ans); exit()**

**print(ans)**

#### **Skipping Stones**

Yogee and his friends are playing skipping stones. Each stone has a value which represents the maximum number of stones you can jump from that particular stone. It was yogee’s turn. Given the value of every stones help him to find the minimum number of jumps required to reach to the last stone provided that he starts from the first stone.

Assume you can always reach the last stone

**Example:-**

Consider that stones have the following value

[2,3,1,3,4]

**Case1:-**

He starts from the first index. a[0]=2,So he can at max reach a[2], a[2]=1 So he can only jump to a[3].a[3]=3 So he can reach end from here with a jump.

Total number of jumps =3

**Case 2:-**

From a[0] he reaches a[1] as he can at maximum jump 2 stones. a[1] is 3,So he can jump 3 stones and can directly reach the end with a jump.

Total number of jumps =2

2 is the expected answer.

**Input format:-**

Total number of stones(n)

Next n lines-Value of stones

**Output format:-**

Minimum number of jumps required.

**Constraints:-**

0<=number of stones<=100000

**ANS.**

**BETTER UNDERSTANDING:**

**n = int(input())**

**stones = [int(input()) for i in range(n)]**

**maxreach = []**

**for i in range(n):**

**if(i+stones[i] >= n-1): maxreach.append(n-1)**

**else: maxreach.append(i+stones[i])**

**i,jumps = 0,0**

**while(i<n):**

**next,mreach = 0,0**

**for j in range(1,stones[i]+1):**

**if(i+j < n-1):**

**if(maxreach[i+j] > mreach):**

**mreach = maxreach[i+j]**

**next = i+j**

**else: next = n; break**

**i = next**

**jumps+=1**

**print(jumps)**

**OPTIMISED:**

**def minjumps(nums,n):**

**if(n == 1 or n == 0): return 0**

**if(nums[0] >= n-1): return 1**

**i,jumps = 0,0**

**while(i<n):**

**next,mreach = 0,0**

**for j in range(1,nums[i]+1):**

**if(i+j < n-1):**

**chk = nums[i+j]+i+j**

**if(chk > mreach): mreach = chk; next = i+j**

**else: next = n; break**

**i = next**

**jumps+=1**

**return jumps**

**n = int(input())**

**nums = [int(input()) for i in range(n)]**

**print(minjumps(nums,n))**

#### **Can they Marry**

Two strings A and B can marry each other  if and only if we can swap any two letters in A so that the result equals B

**Example:-**

A=”ab”, B=”ba”

Yes, As when you swap a and b in A you will get “ba” which is B

**A=”aa”, B=”aa”**

Yes, As when you swap a and a in A you will result in “aa” which is B

**A=”ab”,B=”ab”**

No, As swapping here does not result in B

**Input format:-**

String A

String B

**Output format:-**

Yes or No

**ANS.**

**BRUTE FORCE:**

**a,b = input(),input()**

**if(len(a) != len(b)): print("No")**

**else:**

**test = list(a)**

**for i in range(len(a)-1):**

**for j in range(i+1,len(a)):**

**test[i],test[j] = test[j],test[i]**

**if("".join(test) == b): print("Yes"); exit()**

**test = list(a)**

**print("No")**

**OPTIMISED:**

**a,b = input(),input()**

**if(len(a) != len(b)): print("No"); exit()**

**if(len(set(a))<len(a) and a==b): print("Yes"); exit()**

**chk = [i for i in range(len(a)) if(a[i] != b[i])]**

**if(len(chk)==2 and a[chk[0]]==b[chk[1]] and a[chk[1]]==b[chk[0]]): print("Yes")**

**else: print("No")**

#### **Squares**

Every number can be expressed as sum of perfect squares.

For example

1= 1

So number of perfect squares required 1

2=1+1

So number of perfect squares required 2

3= 1+1+1

So number of perfect squares required 3

4=4

So number of perfect squares required 1

5=1+4

So number of perfect squares required 2

6=4+1+1

So number of perfect squares required 3

12=4+4+4

So number of perfect squares required 3

101=100+1

So number of perfect squares required 2

Given number n find the least number of perfect squares required to represent the sum

**Input format:-**

Number-N

**Output format:-**

Minimum number of perfect squares required.

**Constraints**

1<=N<=2\*106

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <climits>**

**#define long long int**

**using namespace std;**

**void numSquares(int n) {**

**vector<int> dp(n+1,INT\_MAX);**

**vector<int> sq;**

**for(int i=1;i\*i<=n;i++){**

**sq.push\_back(i);**

**}**

**dp[0] = 0;**

**for(int i=1;i<=n;i++){**

**for(int j=0;j<sq.size();j++){**

**if(sq[j]\*sq[j]<=i){**

**dp[i] = min(dp[i-(sq[j]\*sq[j])]+1,dp[i]);**

**}**

**}**

**}**

**cout<<dp[n]<<"\n";**

**}**

**int32\_t main(){**

**int n; cin>>n;**

**numSquares(n);**

**}**

#### **Train routes**

A train stops on n stations numbered from 0 to n - 1 that form a circle. We know the distance between all pairs of neighboring stations where distance[i] is the distance between the station number             i and (i + 1) % n.

The train goes along both directions i.e. clockwise and counterclockwise.

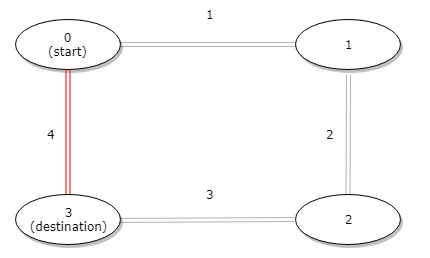
Return the shortest distance between the given start and destination stations.

**Example:-**

**Input:** distance = [1,2,3,4], start = 0, destination = 3

**Output:** 4

**Explanation:**Distance between 0 and 3 is 6 or 4, minimum is 4.

****

**Input format:-**

Number of stations(n)

Next n lines – distances

Starting station

Destination station

**Output format:-**

Minimum distance between start and destination

**ANS.**

**n = int(input())**

**dist = [int(input()) for i in range(n)]**

**start = int(input())**

**end = int(input())**

**d = 0**

**for i in range(end-start):**

**d+=dist[(start+i)%(n-1)]**

**print(min(d,sum(dist)-d))**

#### **Change the word**

You’ll be given with two words.You have to convert the first word to second with minimal number of operation.You can add or delete or replace a character to word1.Find the minimal number of operation.

**Example:-**

**Input:**word1 = "intention", word2 = "execution"

**Output:**5

**Explanation:**

intention -> inention (remove 't')

inention -> enention (replace 'i' with 'e')

enention -> exention (replace 'n' with 'x')

exention -> exection (replace 'n' with 'c')

exection -> execution (insert 'u')

**Input format:-**

First string

Second string

**Output format:-**

Minimal  number of operations

**ANS.**

**a,b = input(),input()**

**dp = [[-1 for i in range(len(b)+1)] for j in range(len(a)+1)]**

**def func(a,b,i,j):**

**if(dp[i][j] != -1): return dp[i][j]**

**if(i == 0):**

**dp[i][j] = j**

**return j**

**if(j == 0):**

**dp[i][j] = i**

**return i**

**if(a[i-1] == b[j-1]):**

**dp[i][j] = func(a,b,i-1,j-1)**

**return dp[i][j]**

**insert = func(a,b,i,j-1)**

**delete = func(a,b,i-1,j)**

**replace = func(a,b,i-1,j-1)**

**dp[i][j] = 1+min(insert,delete,replace)**

**return dp[i][j]**

**print(func(a,b,len(a),len(b)))**

#### **Triplet Sum**

Given an array of numbers find unique triplets such that the sum of the triplets is zero. The solution set must not contain duplicate triplets.Find the number of such triplets in the array

Example:-

Given array nums = [-1, 0, 1, 2, -1, -4],

A solution set is:

[

  [-1, 0, 1],

  [-1, -1, 2]

]

Therefore the required output  is 2

**Input format:-**

Number of elements in an array

Next  line –n elements of array separated by a comma

**Output format:-**

Number of triplets

**ANS.**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**if(n < 3) {cout<<0; exit(0);}**

**string s,num=""; cin>>s; s+=',';**

**vector<int> nums;**

**for(char x: s){**

**if(x == ',') {nums.push\_back(stoi(num)); num = "";}**

**else num+=x;**

**}**

**int cnt = 0;**

**sort(nums.begin(),nums.end());**

**for(int i=0;i<n-2;i++){**

**int l=i+1,r=n-1;**

**while(l<r){**

**int curr\_sum = nums[l]+nums[r]+nums[i];**

**if(curr\_sum == 0) {cnt++; l++; r--;}**

**else if(curr\_sum > 0) r--;**

**else l++;**

**}**

**}**

**cout<<cnt;**

**}**

#### **Cryptography-II**

A message containing letters from A-Z is being encoded to numbers using the following mapping way:'A' -> 1,'B' -> 2,'Z' -> 26

Beyond that, now the encoded string can also contain the character '\*', which can be treated as one of the numbers from 1 to 9.

Given the encoded message containing digits and the character '\*', return the total number of ways to decode it.

For example, 1\* has got 18 decoding possibilities as shown below

(i) 1\* - 1 may be decoded as A and \* can take values from 1 to 9

(ii) 1\* - Keep 1 as 1 and \* can take values from 1 to 9 then characters corressponding to 11 to 19 will be deocoded

**Input Format**

First line contains the encoded string

**Output Format**

Print the decoded string

Also, since the answer may be very large, you should return the output mod 10^9 + 7.

**ANS.**

**#include <iostream>**

**#include <string>**

**#include <vector>**

**#define ll long long**

**using namespace std;**

**ll func(string &s,int ind,int len,vector<ll> &dp){**

**if(ind == len) return 1;**

**if(s[ind] == '0') return 0;**

**if(dp[ind] != -1) return dp[ind];**

**ll res = 0,mul = 1;**

**if(s[ind] == '\*') mul = 9;**

**res += mul\*func(s,ind+1,len,dp)%1000000007;**

**if(ind < len-1){**

**int ops = 0;**

**char next = s[ind+1];**

**if(s[ind] == '\*'){**

**if(next == '\*') ops = 15;**

**else{**

**if(next <= '6') ops = 2;**

**else ops = 1;**

**}**

**}**

**else if(s[ind] < '3'){**

**if(next == '\*'){**

**if(s[ind] == '1') ops = 9;**

**else ops = 6;**

**}**

**else{**

**if(stoi(s.substr(ind,2)) <= 26) ops = 1;**

**}**

**}**

**if(ops) res += ops\*func(s,ind+2,len,dp)%1000000007;**

**}**

**return dp[ind]=res%1000000007;**

**}**

**int main(){**

**string s; cin>>s;**

**int len = s.length();**

**vector<ll> dp(len+1,-1);**

**cout<<func(s,0,len,dp);**

**}**

#### **Chalk sticks**

Ganesh has n chalk sticks of various lengths. He tries to build a square out of the chalk sticks he have. Given the length of n chalk sticks check whether he can build a square or not. Every chalk stick must be used

**Example:-**

**Input:-**[1,1,2,2,2]

**Output:-**Yes

**Explanation:-**As square of side length 2 can be formed from the given lengths

**Example:-**

**Input:**[3,3,3,3,4]

**Output:**No

**Explanation:**You cannot find a way to form a square with all the matchsticks.

**Input format:-**

n-number of chalk sticks

Length of n chalk sticks separated by comma

**Output format:-**

Yes/No

**ANS.**

**n = int(input())**

**s = sorted([int(i) for i in list(input()) if(i != ',')],reverse = True)**

**perimeter\_part = sum(s)/4**

**cnt = 0**

**vis = set()**

**i,p = 0,0**

**while(cnt < 4 and len(s)!=i):**

**if((p+s[i] <= perimeter\_part) and (i not in vis)):**

**p+=s[i]; vis.add(i)**

**if(p == perimeter\_part):**

**p = 0**

**i = 0**

**cnt+=1**

**while(i in vis): i+=1**

**continue**

**i+=1**

**if(cnt == 4): print("Yes")**

**else: print("No")**

#### **Find the number of substrings formed**

Given a string **P** consisting of small English letters and a string **Q** consisting of weight of all characters of English alphabet assigned.

such that for all ‘i’, 0 ≤ Q[i] ≤ 9. The task is to find the total numbers of unique substring that will be formed with sum of weights

atmost **K** value from the given string.

Example:

 Let  P = “abcde”,

the assigned weight for alphabets is

Q = “12345678912345678912345678”,  and K = 5  
 then sub string will be  7 where the sum of weights of individual characters ≤ 5 are:  
“a”, “ab”, “b”, “bc”, “c”, “d”, “e”

**Input Format**

P = “acbacbacaa”

Q = “12300045600078900012345000”

K = 2

**Output Format**

3

**ANS.**

**def wt(p,q):**

**val = 0**

**for char in p:**

**val+=int(q[ord(char)-97])**

**return val**

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

**n = len(p)**

**count = 0**

**vis = set()**

**for i in range(n):**

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

**if(wt(p[i:j],q) <= k and p[i:j] not in vis): count+=1; vis.add(p[i:j])**

**print(count)**

#### **SubArray sum divisible by K**

**Given an array A of integers, return the number of (contiguous, non-empty) subarrays that have a sum divisible by K.**

**Example:-**

**Input: A = [4,5,0,-2,-3,1], K = 5**

**Output: 7**

**Explanation: There are 7 subarrays with a sum divisible by K = 5:**

**[4, 5, 0, -2, -3, 1], [5], [5, 0], [5, 0, -2, -3], [0], [0, -2, -3], [-2, -3]**

**Input:-**

**Number of elements**

**The elements of the array separated by comma**

**K**

**Output format:-**

**No of subarrays whose sum is divisible by K**

**USE SLIDING WINDOWS OF ALL SIZES**

**ANS.**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**vector<int> arr;**

**string numbers; cin>>numbers;**

**string num = "";**

**for(int i=0;i<numbers.length()+1;i++){**

**if(numbers[i] == ',' || numbers[i] == '\0'){**

**arr.push\_back(stoi(num));**

**num = "";**

**continue;**

**}**

**num+=numbers[i];**

**}**

**int k; cin>>k;**

**int sum = 0,cnt = 0;**

**vector<int> window(n);**

**for(int i=0;i<n;i++) {sum+=arr[i]; window[i] = sum;}**

**for(int i=0;i<n;i++){**

**int wnd = window[i],size = i+1;**

**for(int j=size;j<n+1;j++){**

**if(wnd%k == 0) cnt++;**

**wnd+=arr[j]-arr[j-size];**

**}**

**}**

**cout<<cnt;**

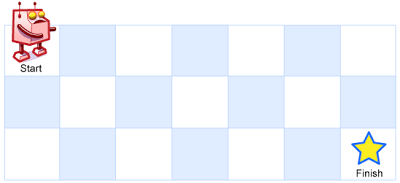
**}**

#### **Unique Path-II**

**A robot is located at the top-left corner of a *m* x *n* grid (marked 'Start' in the diagram below).**

**The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below).**

**Now consider if some obstacles are added to the grids. How many unique paths would there be?**

****

**An obstacle and empty space is marked as 1 and 0 respectively in the grid.**

**Example:-**

**Input:**

**3**

**3**

**[**

**[0,0,0],**

**[0,1,0],**

**[0,0,0]**

**]**

**Output: 2**

**Explanation:**

**There is one obstacle in the middle of the 3x3 grid above.**

**There are two ways to reach the bottom-right corner:**

**1. Right -> Right -> Down -> Down**

**2. Down -> Down -> Right -> Right**

**Input format:-**

**No. of rows**

**No of cols**

**Contents of the array in no.of rows lines separated by comma**

**Output format:- No.of paths possible**

**ANS.**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**bool issafe(vector<vector<char>> &grid,int row,int col,int m,int n){**

**if(row < m && col < n){**

**if(grid[row][col] == '1' || grid[row][col] == 'x') return false;**

**else return true;**

**}**

**return false;**

**}**

**void solve(vector<vector<char>> &grid,int row,int col,int m,int n,int \*cnt){**

**if(row == m-1 && col == n-1){**

**grid[row][col] = 'x';**

**++(\*cnt);**

**// for(auto x: grid){**

**// for(auto y: x){**

**// cout<<y<<" ";**

**// }**

**// cout<<"\n";**

**// }**

**// cout<<"\n";**

**return;**

**}**

**if(col == n) row++;**

**if(issafe(grid,row,col,m,n)){**

**grid[row][col] = 'x';**

**solve(grid,row,col+1,m,n,cnt);**

**solve(grid,row+1,col,m,n,cnt);**

**grid[row][col] = '0';**

**}**

**}**

**int main(){**

**int r,c; cin>>r>>c;**

**vector<vector<char>> grid;**

**for(int i=0;i<r;i++){**

**string c1; cin>>c1;**

**vector<char> row;**

**for(int j=0;j<=c1.length();j++){**

**if(c1[j] != ',' && c1[j] != '\0') row.push\_back(c1[j]);**

**}**

**grid.push\_back(row);**

**}**

**int cnt = 0;**

**solve(grid,0,0,r,c,&cnt);**

**cout<<cnt;**

**}**

#### **Reach the Target**

**A move consists of taking a point (x, y) and transforming it to either (x, x+y) or (x+y, y).**

**Given a starting point (sx, sy) and a target point (tx, ty), print Yes if and only if a sequence of moves exists to transform the point (sx, sy) to (tx, ty). Otherwise, print No**

**Examples:**

**Input: sx = 1, sy = 1, tx = 3, ty = 5**

**Output: Yes**

**Explanation:**

**One series of moves that transforms the starting point to the target is:**

**(1, 1) -> (1, 2)**

**(1, 2) -> (3, 2)**

**(3, 2) -> (3, 5)**

**Input: sx = 1, sy = 1, tx = 2, ty = 2**

**Output: No**

**Input: sx = 1, sy = 1, tx = 1, ty = 1**

**Output: Yes**

**Input format:**

**sx**

**sy**

**tx**

**ty**

**Output format:**

**Yes/No**

**ANS.**

**//BACKTRACKING SOLUTION OR RECURSIVE APPROACH**

**#include <iostream>**

**using namespace std;**

**bool status(int sx,int sy,int tx,int ty){**

**if(sx > tx || sy > ty) return false;**

**if(sx == tx && sy == ty) return true;**

**bool case1 = status(sx+sy,sy,tx,ty);**

**bool case2 = status(sx,sx+sy,tx,ty);**

**return case1|case2;**

**}**

**int main(){**

**int sx,sy,tx,ty; cin>>sx>>sy>>tx>>ty;**

**if(status(sx,sy,tx,ty)) cout<<"Yes";**

**else cout<<"No";**

**}**

#### **All Candidate sum**

**Given a set of candidate numbers (candidates) (without duplicates) and a target number (target), find the number of all unique combinations in candidates where the candidate numbers sums to target.**

**The same repeated number may be chosen from candidates unlimited number of times.**

**Note:**

* **All numbers (including target) will be positive integers.**
* **The solution set must not contain duplicate combinations.**

**Example:-**

**Input: candidates = [2,3,6,7], target = 7,**

**A solution set is:**

**[**

**[7],**

**[2,2,3]**

**]**

**Therefore the expected answer is 2**

**Example :-**

**Input: candidates = [2,3,5], target = 8,**

**A solution set is:**

**[**

**[2,2,2,2],**

**[2,3,3],**

**[3,5]**

**]**

**Therefore the expected answer is 3**

**Input format:-**

**Length of candidate**

**Elements of candidates separated by comma**

**Target sum**

**Output format:-**

**Number of all unique combinations in candidates where the candidate numbers sums to target**

**ANS.**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**for only the count**

**void combination\_sum\_count(vector<int> &arr,int n,int target,int ind,int \*cnt){**

**if(target == 0){**

**++(\*cnt);**

**return;**

**}**

**if(target < 0) return;**

**for(int i=ind;i<n;i++){**

**combination\_sum(arr,n,target-arr[i],i,cnt);**

**}**

**}**

**int main(){**

**int n,ts; cin>>n;**

**vector<int> arr(n);**

**for(int i=0;i<n;i++) cin>>arr[i];**

**cin>>ts;**

**int cnt = 0;**

**combination\_sum(arr,n,ts,0,&cnt);**

**cout<<cnt;**

**}**

#### **First missing positive**

**Given an unsorted integer array, find the smallest missing positive integer.**

**Example 1:**

**Input: [1,2,0]**

**Output: 3**

**Example 2:**

**Input: [3,4,-1,1]**

**Output: 2**

**Example 3:**

**Input: [7,8,9,11,12]**

**Output: 1**

**Note:**

**Your algorithm should run in *O*(*n*)(There should be no nested loops) time and uses constant extra space. You are not supposed to sort the array**

**Input format:-**

**Number of elements**

**Elements separated by comma**

**Output format:-**

**First missing positive**

**ANS.**

**#brute force**

**def binary\_search(arr,k,low,high):**

**while(low <= high):**

**mid = low+(high-low)//2**

**if(arr[mid] == k): return mid**

**else:**

**if(arr[mid] > k): high = mid-1**

**else: low = mid+1**

**return -1**

**n = int(input())**

**arr = list(map(int,input().split(',')))**

**val = 1**

**while(binary\_search(arr,val,0,n-1) != -1): val+=1**

**print(val)**

**#optimised (used extra space)**

**n = int(input())**

**arr = list(map(int,input().split(',')))**

**cycle\_sort = [-1 for i in range(max(arr)+2)]**

**for i in arr:**

**if(i <= 0): continue**

**else: cycle\_sort[i] = i**

**for i in range(1,len(cycle\_sort)):**

**if(cycle\_sort[i] == -1): print(i); exit()**

**print(n+1)**

**#optimised code**

**n = int(input())**

**arr = list(map(int,input().split(',')))**

**i = 0**

**while(i < n):**

**correct = arr[i]-1**

**if(arr[i]<=0 or arr[i]>n or arr[i]==arr[correct]): i+=1**

**else: arr[i],arr[correct] = arr[correct],arr[i]**

**for i in range(len(arr)):**

**if(arr[i] != i+1): print(i+1); exit()**

**print(n+1)**

#### **Maximum width ramp**

**Given an array A of integers, a *ramp* is a tuple (i, j) for which i < j and A[i] <= A[j].  The width of such a ramp is j - i.**

**Find the maximum width of a ramp in A.  If one doesn't exist, return 0.**

**Note: Do not sort the array**

**Example 1:**

**Input: [6,0,8,2,1,5]**

**Output: 4**

**Explanation:**

**The maximum width ramp is achieved at (i, j) = (1, 5): A[1] = 0 and A[5] = 5.**

**Example 2:**

**Input: [9,8,1,0,1,9,4,0,4,1]**

**Output: 7**

**Explanation:**

**The maximum width ramp is achieved at (i, j) = (2, 9): A[2] = 1 and A[9] = 1.**

**Input format:-**

**Number of elements**

**Array elements separated by comma**

**Output format:-**

**Maximum width ramp**

**ANS.**

**BRUTE FORCE (FOR OPTIMISED SEE LEETCODE):**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main(){**

**int n; cin>>n;**

**vector<int> arr;**

**string s; cin>>s;**

**string element = "";**

**for(char c : s+'x'){**

**if(c == ',' || c == 'x'){**

**arr.push\_back(stoi(element));**

**element="";**

**}**

**else element+=c;**

**}**

**int l=0,r=n-1,ans = 0;**

**while(l <= r){**

**int l1=l,r1=r;**

**while(arr[l1] > arr[r]) l1++;**

**while(arr[l] > arr[r1]) r1--;**

**ans = max(ans,max(r1-l,r-l1));**

**l++; r--;**

**}**

**cout<<ans;**

**}**