**Sample output:YES**

**Explanation:**

**Binary value of 5=101**

**Palindrome of 5=101**

**So it is YES else NO**

n=int(input())

a=[]

b=[]

k=0

c=0

while(n>0):

dig=n%2

a.append(dig)

n=n//2

for i in range(len(a)-1,-1,-1):

if a[i]==a[k]:

c=c+1

k=k+1

else:

k=k+1

if(c==len(a)):

print("YES")

else:

print("NO")

**Sample input:**

**2 3**

**2 5 9**

**6 8 3**

**Sample output:**

**1 1 0**

**0 0 1**

**Explanation:**

**The prime number in the matrix is assigned to 1 else it should be assigned to 0**

import numpy as np

R = int(input("Enter the number of rows:"))

C = int(input("Enter the number of columns:"))

k=1

c1=0

print("Enter the entries in a single line (separated by space): ")

# User input of entries in a

# single line separated by space

entries = list(map(int, input().split()))

# For printing the matrix

matrix = np.array(entries).reshape(R, C)

for i in range(0,R):

for j in range(0,C):

while(k<=matrix[i,j]):

if(matrix[i,j]%k==0):

c1+=1

k+=1

else:

k+=1

k=1

if(c1==2):

matrix[i,j]=1

else:

matrix[i,j]=0

c1=0

print(matrix)

Given an even number (greater than 2), return two prime numbers whose sum will be equal to given number. There are several combinations possible. Print only first such pair.

**NOTE:** A solution will always exist, read [Goldbach’s conjecture](https://en.wikipedia.org/wiki/Goldbach's_conjecture). Also, solve the problem in linear time complexity, i.e., O(n).

**Input:**  
The first line contains T, the number of test cases. The following T lines consist of a number each, for which we'll find two prime numbers.

**Note**: The number would always be an even number.

**Output:**  
For every test case print two prime numbers space separated, such that the smaller number appears first. Answer for each test case must be in a new line.

**Constraints:**  
1 ≤ T ≤ 70  
1 ≤ N ≤ 10000

**Example:**  
**Input:**  
5  
74  
1024  
66   
8  
9990

**Output:**  
3 71  
3 1021  
5 61  
3 5  
17 9973

**Program:**

def prime(q):

c=0

for i in range(2,q):

if(q%i==0):

c+=1

break

if(c>=1):

return "false"

else:

return "true"

n=int(input())

a=[]

for i in range(0,n):

a.append(int(input()))

for k in range(2,a[i]):

if(prime(k)=="true" and prime(a[i]-k)=="true"):

print(k,a[i]-k)

break

**Input1:**

5

**Output1:**

1

1 2

1 2 3

1 2 2 2

1 1 1 1 1

**Input2:**

6

**Output2:**

1

1 2

1 2 3

1 2 3 3

1 2 2 2 2

1 1 1 1 1 1

**Program:**

n=int(input())

k=n//2

m=k

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

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

if((n%2)!=0):

if(i>k+1):

if(j>=m):

print(m,end=" ")

else:

print(j,end=" ")

else:

print(j,end=" ")

else:

if(i>k):

if(j>=m):

print(m,end=" ")

else:

print(j,end=" ")

else:

print(j,end=" ")

print()

if((i>k+1) and (n%2)!=0):

m-=1

elif((i>k) and (n%2==0)):

m-=1

**Input**  
3  
aab  
aebaecedabbee  
ab

**Output**  
1  
8  
0

**Explanation  
Testcase 1:**If we change one instance of 'a' to any character from 'c' to 'z', we get all distinct substrings.  
**Testcase 2:** We need to change 2 a's, 2 b's and 4 e's to get distinct substrings.  
**Testcase 3:**As no change is required hence 0.

**Program:**

n=int(input())

a=[]

p=0

for i in range(0,n):

a.append(input())

for j in a:

k=set(j)

s=0

for z in k:

w=j.count(z)

if(w>1):

s=s+(w-1)

print(s)

**Program to remove duplicate characters from the string without using string**

a="aaaaaaabbbbbgggghhhhh"

s=""

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

for j in range(0,i+1):

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

break

if(i==j):

s=s+a[i]

print(s)

**Output:**

abgh

**Zoho problem:**

**input:**

a1b2c3

**output:**

abbccc

**Program:**

strin="a1b2c3"

a=[i for i in strin if i.isalpha()]

b=[]

c=[]

for i in a:

b.append(strin.index(i))

b.append(len(strin))

b=list(map(int,b))

for k in range(0,len(b)-1):

c.append(strin[b[k]+1:b[k+1]])

co=0

for it in a:

for ik in range(0,int(c[co])):

print(it,end="")

co+=1

**Input:**

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

**Output:**

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

**Program:**

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

co=0

b=[]

c=[]

res=[]

for i in a:

if(co%2==0):

b.append(i)

co+=1

else:

c.append(i)

co+=1

b.sort()

b.reverse()

c.sort()

m=min(len(b),len(c))

n=max(len(b),len(c))

for i in range(0,m):

res.append(b[i])

res.append(c[i])

if(n==len(b)):

res.append(b[n-1])

else:

res.append(c[n-1])

print(res)

**Input:**

Periyaiah

**Output:**

p h

e a

r i

i a

y

i a

r i

e a

p h

**Program:**

inp="periyaiah"

a=[[0 for i in range(len(inp))] for j in range(len(inp))]

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

m=len(inp)-1-i

for j in range(0,len(inp)):

if((j==i) or (j==m)):

a[i][j]=inp[j]

else:

a[i][j]=" "

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

for j in range(0,len(inp)):

print(a[i][j],end=" ")

print(" ")

**Program to find the unique elements in the list without using set:**

**Input:**

1,1,2,2,3,3,4,4,5,5,6,6

**Output:**

1,2,3,4,5,6

**Program:**

a=[1,1,2,2,3,3,4,4,5,5,6,6]

b=[]

for i in a:

if i not in b:

b.append(i)

print(b)

* **To check whether the string is palindrome or not without using library function**
* */\**
* *\* C program to find the length of a string without using the*
* *\* built-in function also check whether it is a palindrome*
* *\*/*
* #include <stdio.h>
* #include <string.h>
* void main()
* {
* char string[25], reverse\_string[25] = {'**\0**'};
* int i, length = 0, flag = 0;
* printf("Enter a string **\n**");
* gets(string);
* */\* keep going through each character of the string till its end \*/*
* for (i = 0; string[i] != '**\0**'; i++)
* {
* length++;
* }
* printf("The length of the string '%s' = %d**\n**", string, length);
* for (i = length - 1; i >= 0 ; i--)
* {
* reverse\_string[length - i - 1] = string[i];
* }
* */\* Check if the string is a Palindrome \*/*
* for (flag = 1, i = 0; i < length ; i++)
* {
* if (reverse\_string[i] != string[i])
* flag = 0;
* }
* if (flag == 1)
* printf ("%s is a palindrome **\n**", string);
* else
* printf("%s is not a palindrome **\n**", string);
* }

C Program to Remove given Word from a String

#include <stdio.h>

#include <string.h>

void main()

{

int i, j = 0, k = 0, count = 0;

char str[100], key[20];

char str1[10][20];

printf("enter string:");

scanf("%[^\n]s",str);

/\* Converts the string into 2D array \*/

for (i = 0; str[i]!= '\0'; i++)

{

if (str[i]==' ')

{

str1[k][j] = '\0';

k++;

j = 0;

}

else

{

str1[k][j] = str[i];

j++;

}

}

str1[k][j] = '\0';

printf("enter key:");

scanf("%s", key);

/\* Compares the string with given word \*/

for (i = 0;i < k + 1; i++)

{

if (strcmp(str1[i], key) == 0)

{

for (j = i; j < k + 1; j++)

strcpy(str1[j], str1[j + 1]);

k--;

}

}

for (i = 0;i < k + 1; i++)

{

printf("%s ", str1[i]);

}

}

PROGRAM TO PRINT A PATTERN

Sample Input:

4

Sample Output:

1 2 3 4

2 \* \* 3

3 \* \* 2

4 3 2 1

#include <stdio.h>

int main()

{

int n,l=1,k=1,i,j;

scanf("%d",&n);

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

if(i==1)

{

printf("%d ",j);

}

else if(j==1)

{

printf("%d ",i);

}

else if(j==n && i!=1)

{

printf("%d ",j-l);

l++;

}

else if(i==n && j!=n && j!=1)

{

printf("%d ",i-k);

k++;

}

else

{

printf("\* ");

}

}

printf("\n");

}

}

**Input:**  
2  
i.like.this.program.very.much  
pqr.mno

**Output:**  
i.ekil.siht.margorp.yrev.hcum  
rqp.onm

**Program :**

n=int(input())

a=[]

f=[]

co=0

m=0

c=0

for i in range(0,n):

k=input()

a.append(k)

k=0

for t in range(0,n):

if((a[t].find("."))!=-1):

f.append(1)

else:

f.append(0)

for r in range(0,n):

if(f[r]==0):

print(a[r][::-1])

else:

m=[]

re=[]

m=a[r].split(".")

for b in range(0,len(m)):

re.append(m[b][::-1])

print(".".join(re))

**Program for concentric square pattern:**(for Spiral pattern-https://www.youtube.com/watch?v=O8OIP94OOkU)

**Input:**

4

**Output:**

4 4 4 4 4 4 4

4 3 3 3 3 3 4

4 3 2 2 2 3 4

4 3 2 1 2 3 4

4 3 2 2 2 3 4

4 3 3 3 3 3 4

4 4 4 4 4 4 4

**Program:(https://www.youtube.com/watch?v=rP3U2RAnOkc)**

#include<stdio.h>

int main() {

int n,as,i,j,k,l;

scanf("%d",&n);

as=(n\*2)-1;

int a[as][as];

for(i=0;i<as;i++)

{

for(j=0;j<as;j++)

{

if(abs(i-(as/2))>abs(j-(as/2)))

{

a[i][j]=abs(i-(as/2))+1;

}

else

{

a[i][j]=abs(j-(as/2))+1;

}

}

}

for(k=0;k<as;k++)

{

for(l=0;l<as;l++)

{

printf("%d ",a[k][l]);

}

printf("\n");

}

}

**Smallest greater elements in whole array**:

**Input:**  
2  
9  
6 3 9 8 10 2 1 15 7  
4  
13 6 7 12

**Output**:  
7 6 10 9 15 3 2 \_ 8  
\_ 7 12 13

**Program:**

t=int(input())

b=[]

for i in range(0,t):

b.append(int(input()))

c=[]

d=[]

res=[]

c=list(map(int,input().split(" ")))

d=c[:]

c.sort()

p=set(c)

c=list(p)

c.sort()

for i in d:

if(i==max(c)):

res.append("\_")

else:

h=c.index(i)

res.append(c[h+1])

print(\*res)

c.clear()

d.clear()

res.clear()

**Maximum of all subarrays of size k**

Given an array **A** and an integer **K**. Find the maximum for each and every contiguous subarray of size K.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. The first line of each test case contains a single integer N denoting the size of array and the size of subarray K. The second line contains N space-separated integers A1, A2, ..., ANdenoting the elements of the array.

**Output:**  
Print the maximum for every subarray of size k.

**Constraints:**  
1 ≤ T ≤ 200  
1 ≤ N ≤ 107  
1 ≤ K ≤ N  
0 ≤ A[i] <= 107

**Example:  
Input:**  
2  
9 3  
1 2 3 1 4 5 2 3 6  
10 4  
8 5 10 7 9 4 15 12 90 13

**Output:**  
3 3 4 5 5 5 6  
10 10 10 15 15 90 90

**Explanation:  
Testcase 1:** Starting from first subarray of size k = 3, we have 3 as maximum. Moving the window forward, maximum element are as 3, 4, 5, 5, 5 and 6.

**Program:**

t=int(input())

a=[]

yp=[]

b=[[0 for i in range(t)] for j in range(t)]

for i in range(t):

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

a.append(int(y))

b[i]=list(map(int,input().split()))

for p in range(0,(len(b[i])-a[i])+1):

yp.append(max(b[i][p:p+a[i]]))

print(\*yp)

yp.clear()

**Input:**

100100100

**Output:**

300

**Explanation:**

100+100+100=300

**Program:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String n=sc.nextLine();

int l=n.length();

int a=0;

int sum=0;

while(a!=l)

{

sum=sum+Integer.parseInt(n.substring(a,a+3));

a+=3;

}

System.out.println(sum);

}

}

**Longest palindrome in a string:**

**Input:**

forgeeksskeegfor

**Output:**

geeksskeeg

**Program:**

s=input()

a=[]

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

k=len(s)-1

while(k>=i+1):

if(s[i]==s[k]):

if(i==0):

if(s[i:k+1]==s[k::-1]):

a.append(s[i:k+1])

break

else:

if(s[i:k+1]==s[k:i-1:-1]):

a.append(s[i:k+1])

break

k-=1

c=[]

b=[]

if(len(a)==0):

print("no")

else:

for e in a:

c.append(len(e))

for i in a:

if(len(i)==max(c)):

b.append(i)

print(\*b)

**Paranthesis checker:**

**Input:**

{()}[]

**Output:**

Balanced

**Program:**

class stack:

def \_\_init\_\_(self):

self.elements=[]

def check(self,li,br,lo):

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

if(br[i] in li):

self.elements.append(br[i])

else:

h=lo.index(br[i])

if(li[h] in self.elements):

self.elements.remove(li[h])

if(len(self.elements)==0):

return 1

else:

return 0

br=input()

s=stack()

li=['{','(','[']

lo=['}',')',']']

if(s.check(li,br,lo)==1):

print("balanced")

else:

print("unbalanced")

**New Year Chaos**

**Sample Input**

2

5

2 1 5 3 4

5

2 5 1 3 4

**Sample Output**

3

Too chaotic

**Explanation**

**Test Case 1**

The initial state:



After person 5 moves one position ahead by bribing person 4:



Now person 5 moves another position ahead by bribing person 3:



And person 2 moves one position ahead by bribing person 1:



So the final state is 2,1,5,3,4 after three bribing operations.

**Test Case 2**

No person can bribe more than two people, so its not possible to achieve the input state.

**Program:**

import math

import os

import random

import re

import sys

# Complete the minimumBribes function below.

def minimumBribes(q):

    i=len(q)

    c=0

    f=0

    while((i>0) and (f==0)):

        if(i!=q[i-1]):

            if(i==q[i-2]):

                c=c+1

                t=q[i-2]

                q[i-2]=q[i-1]

                q[i-1]=t

            elif(i==q[i-3]):

                c=c+2

                t=q[i-3]

                q[i-3]=q[i-2]

                q[i-2]=t

                t=q[i-2]

                q[i-2]=q[i-1]

                q[i-1]=t

            else:

                print("Too chaotic")

                f==1

                c=0

                break

        i-=1

    if((f==0) and (c>0)):

        print(c)

if \_\_name\_\_ == '\_\_main\_\_':

    t = int(input())

    for t\_itr in range(t):

        n = int(input())

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

        minimumBribes(q)

**Input:**

7 21

**Output:**

---------.|.---------

------.|..|..|.------

---.|..|..|..|..|.---

-------WELCOME-------

---.|..|..|..|..|.---

------.|..|..|.------

---------.|.---------

**Program:**

r,c=map(int,input().split())

mat=[]

dash=(c-3)//2

odot='.'

edot=".."

sl="|"

ced=0

ty=r-1

for i in range(0,(r//2)):

    st=""

    k=""

    for j in range(dash):

        k=k+"-"

    li=""

    for i in range(0,ced):

        li=li+sl+edot

    li=li+sl

    st=st+k+odot+li+odot+k

    mat.append(st)

    ced+=2

    dash-=3

ru=(c-7)//2

pr=""

for i in range(0,ru):

    pr=pr+"-"

wq=pr+"WELCOME"+pr

mat.append(wq)

for po in range((r//2)-1,-1,-1):

    mat.append(mat[po])

for pt in mat:

    print(pt)

**L Matrix and inverted L Matrix:**

**Sample input:**

6

1 3 5 7 9 11

2 4 6 8 10 12

1 3 5 7 9 11

2 4 6 8 10 12

1 3 5 7 9 11

2 4 6 8 10 12

**Sample output:**

11 9 7 5 3 1 2 1 2 1 2

12 11 12 11 12 10 8 6 4

10 8 6 4 3 4 3

9 10 9 7 5

7 5 6

8

**Program:**

n=int(input())

mat=[]

for i in range(n):

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

mat.append(a)

kl=(2\*n)-1

r=0

c=n-1

ps=0

for j in range(n):

tmpr=r

tmpc=c

psk=kl

for k in range(psk):

if(k<=psk//2):

print(mat[tmpr][tmpc],end=" ")

if(ps==0):

if(k!=psk//2):

tmpc-=1

if(ps==1):

if(k!=psk//2):

tmpr+=1

if(k>psk//2):

if(ps==0):

if(k!=psk):

tmpr+=1

if(ps==1):

if(k!=psk):

tmpc-=1

print(mat[tmpr][tmpc],end=" ")

print()

if(ps==0):

ps=1

r+=1

else:

ps=0

c-=1

kl-=2

Integer diagonally:

Input:

4 4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

11 1

Output:

YES

Program:

r,c=map(int,input().split())

mat=[]

for jk in range(r):

kl=list(map(int,input().split()))

mat.append(kl)

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

fl=0

for kp in range(r):

mk=0

try:

q=mat[kp].index(x)

p=kp

mk=1

except:

pass

if(mk==1):

tmp1=p

tmp2=q

var1=q

while(((tmp1>=0) and (tmp1<=r-1)) and ((tmp2>=0) and (tmp2<=c-1)) and (fl==0)):

if(mat[tmp1][tmp2]==y):

fl=1

if(mat[tmp1][var1]==y):

fl=1

tmp1+=1

tmp2+=1

var1-=1

if(fl==0):

while((tmp1>=0) and (var1<=c-1) and (tmp1<=r-1) and (var1>=0)):

if(mat[tmp1][var1]==y):

fl=1

tmp1+=1

var1-=1

tmp1=p

tmp2=q

var2=q

while(((tmp1>=0) and (tmp1<=r-1)) and ((tmp2>=0) and (tmp2<=c-1)) and (fl==0)):

if(mat[tmp1][tmp2]==y):

fl=1

if(mat[tmp1][var2]==y):

fl=1

tmp1-=1

tmp2-=1

var2+=1

if(fl==0):

while((tmp1>=0) and (var2<=c-1) and (tmp1<=r-1) and (var2>=0)):

if(mat[tmp1][var2]==y):

fl=1

tmp1-=1

var2+=1

if(fl==1):

print("YES")

else:

print("NO")

Problem

You are given an integer such that the XOR of two integers is . In short (⊕ denotes the bitwise the XOR operation).

Out of all possible pairs of and , you must find two integers such that their product is maximum.

Let us define as the length of in its binary representation. Then, and .

Input format

A single integer representing ()

Output format

Print the maximum product you can achieve under the given conditions.

Sample Input

13

Sample Output

70

Explanation:

The binary representation of 13 is "1101".

We can use A=10 ("1010" in binary) and B=7 ("0111" in binary). This gives us the product 70. No other valid pair (A,B) can give us a larger product.

Program:

n=int(input())

arr=[]

while(n>0):

arr.insert(0,n%2)

n=n//2

m1=0

m2=0

vs=len(arr)

for mg in range(vs):

if(arr[mg]==1):

if(m1<=m2):

m1+=(2\*\*(vs-1-mg))

else:

m2+=(2\*\*(vs-1-mg))

else:

m1+=(2\*\*(vs-1-mg))

m2+=(2\*\*(vs-1-mg))

print(m1\*m2)

Sample input:

eeddccbbaajklmnop

Sample output:

eeddccbbaa

Program:

st=input()

li=[]

ans=""

for ptr in range(1,len(st)):

if(st[ptr-1]>=st[ptr]):

ans+=st[ptr-1]

else:

ans+=st[ptr-1]

li.append(ans)

ans=""

li.append(ans)

if(st[-2]>=st[-1]):

li[-1]=li[-1]+st[-1]

else:

li[-1]=st[-1]

print(li)