# HackerRank programs

# Encryption

#!/bin/python3

import math

import os

import random

import re

import sys

# Complete the encryption function below.

def encryption(s):

    s=s.replace(" ","")

    l=len(s)

    r=math.floor(math.sqrt(l))

    c=math.ceil(math.sqrt(l))

    if r\*c<l: r+=1

    k=0

    m=[]

    for i in range (r):

        a=[]

        for j in range(c):

            if k<l:

                a.append(s[k])

                k+=1

        m.append(a)

    str=""

    for j in range(c):

        for i in range(r):

            try:

                str+=m[i][j]

            except:

                continue

        str+=" "

    str=str.strip()

    return str

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    s = input()

    result = encryption(s)

    fptr.write(result + '\n')

    fptr.close()

# Queen's Attack II

#!/bin/python3

import math

import os

import random

import re

import sys

# Complete the queensAttack function below.

def queensAttack(n, k, r\_q, c\_q, obstacles):

    count=0

    i=r\_q+1

    j=c\_q

    while i<=n:

        if [i,j] not in obstacles:

            count+=1

            i+=1

        else: break

    i=r\_q-1

    j=c\_q

    while i>=1:

        if [i,j] not in obstacles:

            count+=1

            i-=1

        else: break

    i=r\_q

    j=c\_q+1

    while j<=n:

        if [i,j] not in obstacles:

            count+=1

            j+=1

        else: break

    i=r\_q

    j=c\_q-1

    while j>=1:

        if [i,j] not in obstacles:

            count+=1

            j-=1

        else: break

    i=r\_q+1

    j=c\_q+1

    while i<=n and j<=n:

        if [i,j] not in obstacles:

            count+=1

            i+=1

            j+=1

        else: break

    i=r\_q-1

    j=c\_q-1

    while i>=1 and j>=1:

        if [i,j] not in obstacles:

            count+=1

            i-=1

            j-=1

        else: break

    i=r\_q+1

    j=c\_q-1

    while i<=n and j>=1:

        if [i,j] not in obstacles:

            count+=1

            i+=1

            j-=1

        else: break

    i=r\_q-1

    j=c\_q+1

    while i>=1 and j<=n:

        if [i,j] not in obstacles:

            count+=1

            i-=1

            j+=1

        else: break

    return count

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    nk = input().split()

    n = int(nk[0])

    k = int(nk[1])

    r\_qC\_q = input().split()

    r\_q = int(r\_qC\_q[0])

    c\_q = int(r\_qC\_q[1])

    obstacles = []

    for \_ in range(k):

        obstacles.append(list(map(int, input().rstrip().split())))

    result = queensAttack(n, k, r\_q, c\_q, obstacles)

    fptr.write(str(result) + '\n')

    fptr.close()

**Climbing the Leaderboard**

#!/bin/python3

import math

import os

import random

import re

import sys

# Complete the climbingLeaderboard function below.

def climbingLeaderboard(scores, alice):

    scores=list(dict.fromkeys(scores))

    res=[]

    for i in alice:

        for j in scores[::-1]:

            if i<j:

                res.append(scores.index(j)+2)

                break

            elif i==j:

                res.append(scores.index(j)+1)

                break

            elif i>scores[0]:

                res.append(1)

                break

    return res

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    scores\_count = int(input())

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

    alice\_count = int(input())

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

    result = climbingLeaderboard(scores, alice)

    fptr.write('\n'.join(map(str, result)))

    fptr.write('\n')

    fptr.close()

# Extra Long Factorials

#!/bin/python3

import math

import os

import random

import re

import sys

# Complete the extraLongFactorials function below.

def extraLongFactorials(n):

    fact=1

    while(n):

        fact\*=n

        n-=1

    print(fact)

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    extraLongFactorials(n)

# Fraudulent Activity Notifications

#!/bin/python3

import math

import os

import random

import re

import sys

# Complete the activityNotifications function below.

def activityNotifications(expenditure, d):

    count=0

    r=d

    while(r<n):

        a=[]

        for i in range(r-d,r):

            a.append(expenditure[i])

        a.sort()

        if d%2==0:

            med=(a[d//2-1]+a[d//2])/2

        else:

            med=a[d//2]

        if(expenditure[d]>=2\*med): count+=1

        r+=1

    return count

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    nd = input().split()

    n = int(nd[0])

    d = int(nd[1])

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

    result = activityNotifications(expenditure, d)

    fptr.write(str(result) + '\n')

    fptr.close()

# Bigger is Greater

#!/bin/python3

import math

import os

import random

import re

import sys

# Complete the biggerIsGreater function below.

def biggerIsGreater(w):

    l=len(w)

    for i in range(l-1,0,-1):

        if w[i]>w[i-1]:

            if i==l-1:

                w=list(w)

                w[i],w[l-2]=w[l-2],w[i]

                w=''.join(w)

                return (w)

            else:

                w=list(w)

                w[i],w[l-1]=w[l-1],w[i]

                w[i-1],w[l-2]=w[l-2],w[i-1]

                w=''.join(w)

                return(w)

    return('no answer')

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    T = int(input())

    for T\_itr in range(T):

        w = input()

        result = biggerIsGreater(w)

        fptr.write(result + '\n')

    fptr.close()

# The Time in Words

#!/bin/python3

import math

import os

import random

import re

import sys

# Complete the timeInWords function below.

def timeInWords(h, m):

    a=["zero", "one","two","three","four","five","six","seven","eight","nine","ten","eleven","twelve","thirteen","fourteen","quarter","sixteen","seventeen","eighteen","nineteen","twenty","twenty one","twenty two","twenty three","twenty four","twenty five","twenty six","twenty seven","twenty eight","twenty nine","half"]

    t=a[h]

    if(m==0): t=t+" o' clock"

    elif(m==1): t=a[m]+" minute past "+t

    elif(m==15 or m==30): t=a[m]+" past "+t

    elif(m<30): t=a[m]+" minutes past "+t

    else:

        m=60-m

        if(h==12): h=0

        if(m==15): t=a[m]+" to "+a[h+1]

        else:t=a[m]+" minutes to "+a[h+1]

    return(t)

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    h = int(input())

    m = int(input())

    result = timeInWords(h, m)

    fptr.write(result + '\n')

    fptr.close()

# The Full Counting Sort

#!/bin/python3

import math

import os

import random

import re

import sys

def num(arr):

    return arr[0]

# Complete the countSort function below.

def countSort(arr):

    for i in range(n):

        arr[i][0]=int(arr[i][0])

    for i in range (n//2):

        arr[i][1]='-'

    arr.sort(key=num)

    for i in range(n):

        print(arr[i][1],end=" ")

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input().strip())

    arr = []

    for \_ in range(n):

        arr.append(input().rstrip().split())

    countSort(arr)