Name-Priyanshu Singh

Assignment Day-1

Albanero

Question 1:

Q1. Generate a String With Characters That Have Odd Counts

Given an integer n, return a string with n characters such that each character in such string occurs an odd number of times.

The returned string must contain only lowercase English letters. If there are multiples valid strings, return any of them.

Example 1:

Input: n = 4

Output: "pppz"

Explanation: "pppz" is a valid string since the character 'p' occurs three times and the character 'z' occurs once. Note that there are many other valid strings such as "ohhh" and "love".

Example 2:

Input: n = 2

Output: "xy"

Explanation: "xy" is a valid string since the characters 'x' and 'y' occur once. Note that there are many other valid strings such as "ag" and "ur".

Example 3:

Input: n = 7

Output: "holasss"

Constraints:

1 <= n <= 500

Code:

a=int(input())

if a%2==0:

for i in range(a-1):

print('x',end="")

print('y')

else:

for i in range(a):

print('s',end="")

Question 2:

Q2.Maximum Nesting Depth of the Parentheses

A string is a valid parentheses string (denoted VPS) if it meets one of the following:

It is an empty string "", or a single character not equal to "(" or ")",

It can be written as AB (A concatenated with B), where A and B are VPS's, or

It can be written as (A), where A is a VPS.

We can similarly define the nesting depth depth(S) of any VPS S as follows:

depth("") = 0

depth(C) = 0, where C is a string with a single character not equal to "(" or ")".

depth(A + B) = max(depth(A), depth(B)), where A and B are VPS's.

depth("(" + A + ")") = 1 + depth(A), where A is a VPS.

For example, "", "()()", and "()(()())" are VPS's (with nesting depths 0, 1, and 2), and ")(" and "(()" are not VPS's.

Given a VPS represented as string s, return the nesting depth of s.

Example 1:

Input: s = "(1+(2\*3)+((8)/4))+1"

Output: 3

Explanation: Digit 8 is inside of 3 nested parentheses in the string.

Example 2:

Input: s = "(1)+((2))+(((3)))"

Output: 3

Example 3:

Input: s = "1+(2\*3)/(2-1)"

Output: 1

Example 4:

Input: s = "1"

Output: 0

Constraints:

1 <= s.length <= 100

s consists of digits 0-9 and characters '+', '-', '\*', '/', '(', and ')'.

It is guaranteed that parentheses expression s is a VPS.

Code:

a=input()

from queue import LifoQueue

s=LifoQueue()

max11=0

for b in a:

if(b=='('):

s.put(b)

max11=max(max11,s.qsize())

elif(b==')'):

s.get()

print(max11)

Question 3:

Q3. Given an array of integers, determine whether the sum of its elements is even or odd. The output should be a string ("odd" or "even").

  If the input array is empty, consider it as an array with a zero ([0]).

  Examples

    evenOrOdd([0]) ➞ "even"

    evenOrOdd([1]) ➞ "odd"

    evenOrOdd([]) ➞ "even"

    evenOrOdd([0, 1, 5]) ➞ "even"

Code:

from array import\*

a=int(input())

s=array('i',[])

maz,q=0,0

for q in range(a):

s.append(int(input()))

maz=maz+s[q]

q=q+1

if maz %2==0:

print("even",maz)

else:

print("odd",maz)

Question 4:

Q4. Add 1 to every element in the given array.

  Examples

  [0, 1, 2, 3] ➞ [1, 2, 3, 4]

  [2, 4, 6, 8] ➞ [3, 5, 7, 9]

  [-1, -2, -3, -4] ➞ [0, -1, -2, -3]

Code:

a=int(input())

from array import\*

s=array('i',[])

for q in range(a):

s.append(int(input())+1)

for w in s:

print(w)

Question 5:

Q5. Given an array of either entirely odd integers or entirely even integers except for a single Outlier Number. Write a program to return this number.

  Examples

    [2, 3, 4] ➞ 3

    // 2 and 4 are even numbers.

    // 3 is an outlier number.

    [1, 2, 3] ➞ 2

    [4, 1, 3, 5, 9] ➞ 4

Code:

a=int(input())

from array import\*

s=array('i',[])

l,p=0,0

q=0

q1=0

for q in range(a):

s.append(int(input()))

for r in s:

if(r%2==0):

l+=1

q=r

else:

p+=1

q1=r

if(l==1):

print(q)

else:

print(q1)

Question 6:

Q6. In mathematics, an Arithmetic Progression (AP) is a sequence of numbers such that the difference between the consecutive terms is constant. Write a program that takes three arguments:

  The first element of the sequence, first

  Constant difference between the elements, diff

  Total numbers in the sequence, n

  Return the first n elements of the sequence with the given common difference diff. Final result should be a string of numbers, separated by comma and space.

  Examples

    first=1, diff=2, n=5 ➞ "1, 3, 5, 7, 9"

    first=1, diff=0, n=5 ➞ "1, 1, 1, 1, 1"

    first=1, diff=-3,n=10 ➞ "1, -2, -5, -8, -11, -14, -17, -20, -23, -26"

Code:

a=int(input())

d=int(input())

n=int(input())

s=0

p=""

for x in range(n):

if(x==0):

s=s+a

else:

s=s+d

p=p+str(s)

if(x!=n-1):

p=p+','

print(p)

Question 7:

Q7. Given an array A of 0s and 1s, we may change up to K values from 0 to 1.

Return the length of the longest (contiguous) subarray that contains only 1s.

Example 1:

Input: A = [1,1,1,0,0,0,1,1,1,1,0], K = 2

Output: 6

Explanation:

[1,1,1,0,0,1,1,1,1,1,1]

Bolded numbers were flipped from 0 to 1.  The longest subarray is underlined.

Example 2:

Input: A = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], K = 3

Output: 10

Explanation:

[0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1]

Bolded numbers were flipped from 0 to 1.  The longest subarray is underlined.

Code:

a=int(input())

K=int(input())

l=list()

l1=list()

for s in range(a):

l.append(int(input()))

for s in range(a):

if(l[s]==0):

l1.append(s)

f,q=0,0

maxx1=0

d=0

s=0

while(s<a):

if(l[s]==0):

f=f+1

if(f>K):

maxx=q+K

maxx1=max(maxx,maxx1)

f=K-1

s=s-1

i=l1[d]

while(i>=0):

if(l[i]==1):

q-=1

i=i-1

d=d+1

else:

q=q+1

s=s+1

if(len(l1)==0 or len(l1)<K):

maxx1=a

elif(f<=K):

maxx1=max(maxx1,q+K)

print(maxx1)

Question 8:

Q8. Given an array A of integers, we must modify the array in the following way: we choose an i and replace A[i] with -A[i], and we repeat this process K times in total.  (We may choose the same index i multiple times.)

Return the largest possible sum of the array after modifying it in this way.

Example 1:

Input: A = [4,2,3], K = 1

Output: 5

Explanation: Choose indices (1,) and A becomes [4,-2,3].

Example 2:

Input: A = [3,-1,0,2], K = 3

Output: 6

Explanation: Choose indices (1, 2, 2) and A becomes [3,1,0,2].

Example 3:

Input: A = [2,-3,-1,5,-4], K = 2

Output: 13

Explanation: Choose indices (1, 4) and A becomes [2,3,-1,5,4].

Code:

a=int(input())

k=int(input())

u=0

s=list()

for b in range(a):

s.append(int(input()))

s.sort()

#print(s)

for b in range(a):

if(s[b]<0 and u<k):

s[b]=s[b]\*-1

u+=1

p=min(s)

#print(s)

k1=p

while(u<k):

p=p\*-1

u+=1

l=0

for x in s:

l=l+x

print(l-k1+p)

Question 9:

**Q9**. You are given an array prices where prices[i] is the price of a given stock on the ith day.

Find the maximum profit you can achieve. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times).

Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

Example 1:

Input: prices = [7,1,5,3,6,4]

Output: 7

Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.

Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.

Example 2:

Input: prices = [1,2,3,4,5]

Output: 4

Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.

Note that you cannot buy on day 1, buy on day 2 and sell them later, as you are engaging multiple transactions at the same time. You must sell before buying again.

Example 3:

Input: prices = [7,6,4,3,1]

Output: 0

Explanation: In this case, no transaction is done, i.e., max profit = 0.

Code:

a=int(input())

l=list()

for a1 in range(a):

l.append(int(input()))

f=0

r=0

p=0

for aa in range(a):

if(f==0 and aa!=a-1):

if(l[aa]<l[aa+1]):

r=l[aa]

f=1

elif(aa==a-1 or l[aa]>l[aa+1]):

f=0

p=p+l[aa]-r

print(p)

Question 10:

Q10. Take an input, array of numbers and return only the even values.

  Examples

    [1, 2, 3, 4, 5, 6, 7, 8] ➞ [2, 4, 6, 8]

    [43, 65, 23, 89, 53, 9, 6] ➞ [6]

    [718, 991, 449, 644, 380, 440] ➞ [718, 644, 380, 440]

  Notes

    Return all even numbers in the order they were given.

    All test cases contain valid numbers ranging from 1 to 3000.

Code:

a=int(input())

l=list()

for i in range(a):

l.append(int(input()))

ll=list()

for j in range(a):

if(l[j]%2==0):

ll.append(l[j])

print(ll)