Program-1:

Input: Given a List of numbers separated with comma.

The numbers 5 and 8 are present in the list.

Assume that 8 always come after 5.

Case 1: num1 -> Add all numbers which do not lie between 5 and 8 in the Input List.

Case 2: num2 -> Numbers formed by concatenating all numbers from 5 to 8 in the list.

Output: Sum of num1 and num2

Example: 3,2,6,5,1,4,8,9

Num1: 3+2+6+9 =20

Num2: 5148

O/p=5148+20 = 5168

```
L=list(map(int,input().split(",")))
flag=True
s=0
s1=0
#3,2,6,5,1,4,8,9
for i in L:
    if(i==5 or i==8):
        flag= not flag
        s1=s1*10+i
        continue
    if(flag):
        s=s+i

        continue
    s1=s1*10+i
print(s+s1)
```

Program-2:

A string which is a mixture of letters ,numbers and special characters from which produce the largest even number from the available digit after removing the duplicates digits.

If an even number did not produce then return -1.

Ex : infosys@337

O/p : -1

Hello#81@21349

O/p :983412

```
s=input()
R=""
for i in s:
    if(ord(i)>=48 and ord(i)<=57):
        R+=i
if("2" in R or "4" in R or"6" in R or"8" in R or"0" in R):
    L=list(R)
    L=list(set(L))
    L.sort(reverse=True)
    for i in L[::-1]:
        if(int(i)%2==0):
            t=i
            break
    L.remove(t)
    L.append(t)
    print("".join(L))
else:
    print(-1)
```

Program-3:

Given m*n matrix where m defines number of rows and n defines number of columns. Your job is to check whether a number is consecutive for 3 times either in row, column, or diagonal. If there are more than one such numbers then print the minimum one.

Note : n = m+1

Input :

First Input : m, displaying number of rows

Second Input : m*n matrix

```
2 3 4 5 6 2 4 3
2 3 4 7 6 7 6 2
2 3 5 5 5 5 2 5
2 3 1 1 2 1 3 6
1 1 1 1 9 0 3 5
2 3 1 1 5 1 2 7
```

O/p :1

```
r=int(input())
c=int(input())
L=[]
for i in range(r):
    L.append(list(map(int,input().split())))
m=float('inf')
for i in range(r-2):

    for j in range(c-2):
        if(m>L[i][j] and L[i][j]==L[i][j+1] and L[i][j]==L[i][j+2]):
            m=L[i][j]
        if(m>L[i][j] and L[i][j]==L[i+1][j] and L[i][j]==L[i+2][j]):
            m=L[i][j]
        if(m>L[i][j] and L[i][j]==L[i+1][j+1] and L[i][j]==L[i+2][j+2]):
            m=L[i][j]
print(m)
```

Program-4:

Take input a number 'N' and an array as given below.

Input:- N=2

Array =1,2,3,3,4,4

O/p : 2

Find the least number of unique elements after deleting N numbers of elements from the array.

In the above example, after deleting N=2 elements from the array.

In above 1,2 will be deleted.

So 3,3,4,4 will be remaining so,

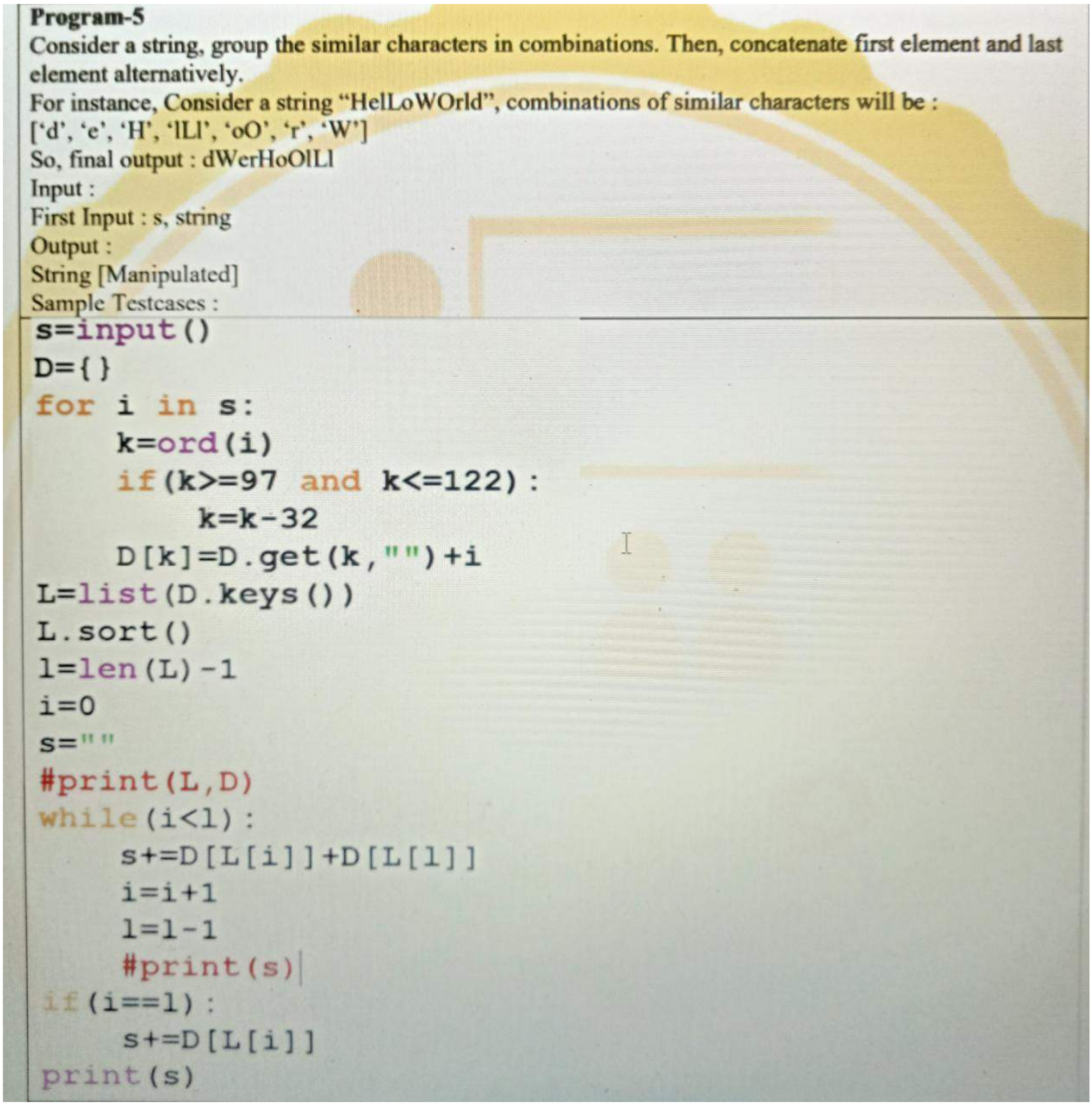
2 unique elements are in the array i.e 3 and 4.

So ,output will be 2.

```
L=list(map(int,input().split()))
N=int(input())
D={}
for i in L:
    D[i]=D.get(i,0)+1
L=list(D.values())
L.sort()
i=0
while(N>0):
    N=N-L[i]
    i=i+1

if(N<0):
    i=i-1

print(len(L)-i)
```



Program-5

Consider a string, group the similar characters in combinations. Then, concatenate first element and last element alternatively.

For instance, Consider a string "HelLoWOrld", combinations of similar characters will be :

['d', 'e', 'H', 'lLl', 'oO', 'r', 'W']

So, final output : dWerHoOlLl

Input :

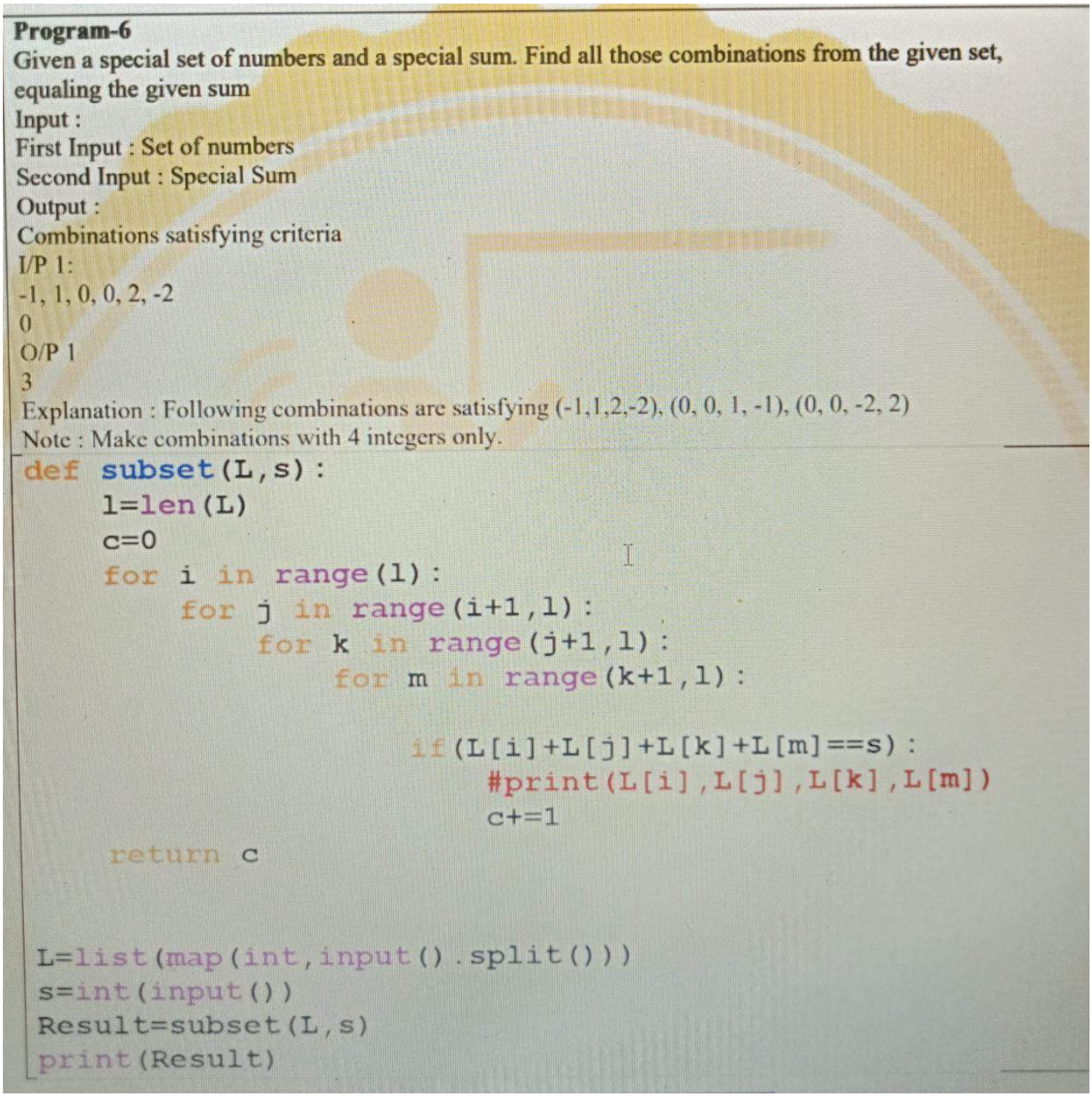
First Input : s, string

Output :

String [Manipulated]

Sample Testcases :

```
s=input()
D={}
for i in s:
    k=ord(i)
    if(k>=97 and k<=122):
        k=k-32
    D[k]=D.get(k,"")+i
L=list(D.keys())
L.sort()
l=len(L)-1
i=0
s=""
#print(L,D)
while(i<l):
    s+=D[L[i]]+D[L[l]]
    i=i+1
    l=l-1
    #print(s)
if(i==l):
    s+=D[L[i]]
print(s)
```



Program-6

Given a special set of numbers and a special sum. Find all those combinations from the given set, equaling the given sum

Input :

First Input : Set of numbers

Second Input : Special Sum

Output :

Combinations satisfying criteria

I/P 1:

-1, 1, 0, 0, 2, -2

0

O/P 1

3

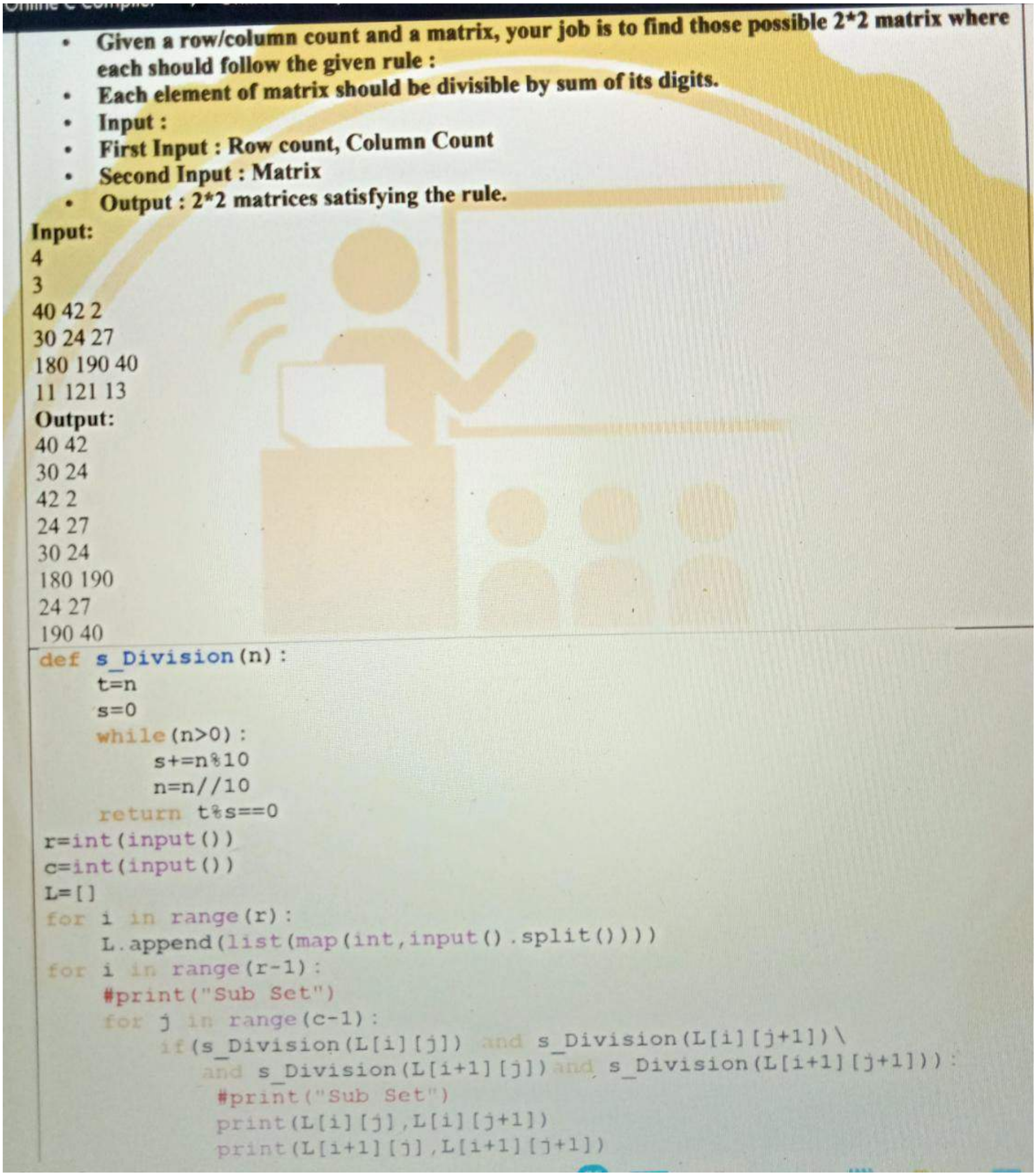
Explanation : Following combinations are satisfying (-1,1,2,-2), (0, 0, 1, -1), (0, 0, -2, 2)

Note : Make combinations with 4 integers only.

```
def subset(L,s):
    l=len(L)
    c=0
    for i in range(l):
        for j in range(i+1,l):
            for k in range(j+1,l):
                for m in range(k+1,l):

                    if(L[i]+L[j]+L[k]+L[m]==s):
                        #print(L[i],L[j],L[k],L[m])
                        c+=1
    return c


L=list(map(int,input().split()))
s=int(input())
Result=subset(L,s)
print(Result)
```



- **Given a row/column count and a matrix, your job is to find those possible 2*2 matrix where each should follow the given rule :**
- **Each element of matrix should be divisible by sum of its digits.**
- **Input :**
- **First Input : Row count, Column Count**
- **Second Input : Matrix**
- **Output : 2*2 matrices satisfying the rule.**

Input:

```
4
3
40 42 2
30 24 27
180 190 40
11 121 13
```

Output:

```
40 42
30 24
42 2
24 27
30 24
180 190
24 27
190 40
```

```python
def s_Division(n):
    t=n
    s=0
    while(n>0):
        s+=n%10
        n=n//10
    return t%s==0
r=int(input())
c=int(input())
L=[]
for i in range(r):
    L.append(list(map(int,input().split())))
for i in range(r-1):
    #print("Sub Set")
    for j in range(c-1):
        if(s_Division(L[i][j]) and s_Division(L[i][j+1])\
          and s_Division(L[i+1][j]) and s_Division(L[i+1][j+1])):
            #print("Sub Set")
            print(L[i][j],L[i][j+1])
            print(L[i+1][j],L[i+1][j+1])
```

Program-8:

String Rotation

Input :- rhdt:246,ghftd:1246

Output:- trdt,ftdgh

Explanation :here every string is associated with the number separated with ':' if sum of squares of digits is even then rotate the string left by 1 position or if sum of squares of digits is odd then rotate the string right by 2 position.

2*2+4*4+6*6=56 which is even so rotate left by 1 position rhdt --->trhd.

1*1+2*2+4*4+6*6=57 which is odd then rotate right by 2 position ghftd---> ftdgh

```
L=list(input().split(","))
for i in L:
    t=i.split(":")
    k=0
    for j in t[1]:
        k+=int(j)*int(j)
    if(k%2==0):
        print(t[0][-1]+t[0][:-1])
    else:
        print(t[0][2:]+t[0][:2])
```

Program-9

Given an array of n elements. Create a square matrix and Print all sub matrix whose sum is highest.

Input :

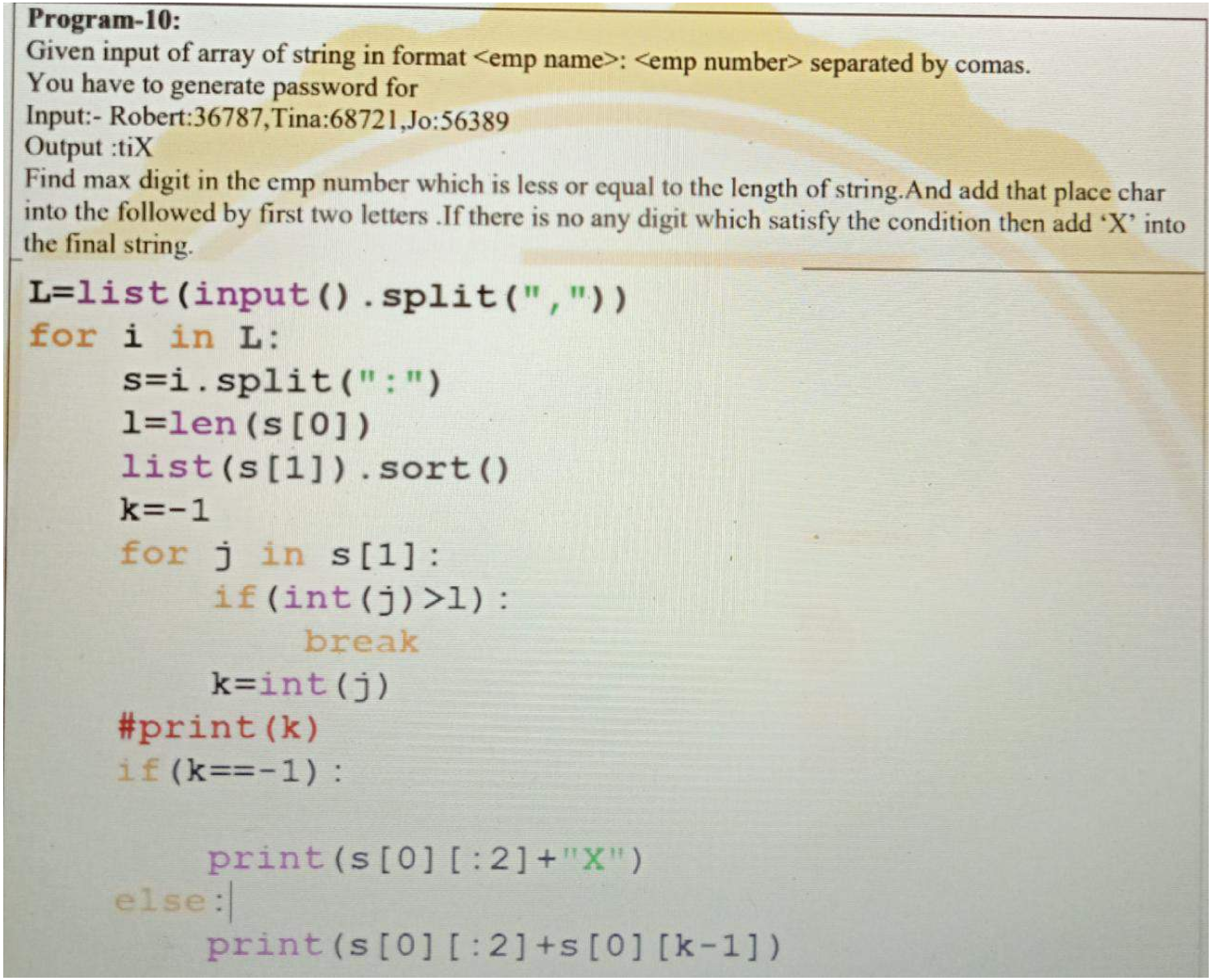
6,3,6,20,3,6,-15,3,3

Output:

```
6 3 6
20 3 6
-15 3 3
Sum : 35
6 3
6 20
Sum= 35
6 20
3 6
Sum=35
```

```python
import math
L=list(map(int,input().split(",")))
D={}
l=len(L)
r=int(math.sqrt(l))
while(r>0):
    for i in range(l-r*r+1):
        s=sum(L[i:i+r*r])
        if(s in D.keys()):
            D[s].append(L[i:i+r*r])
        else:
            D[s]=[]
            D[s].append(L[i:i+r*r])


    r-=1
l=D[max(D.keys())]
for i in l:
    print(*i)
```

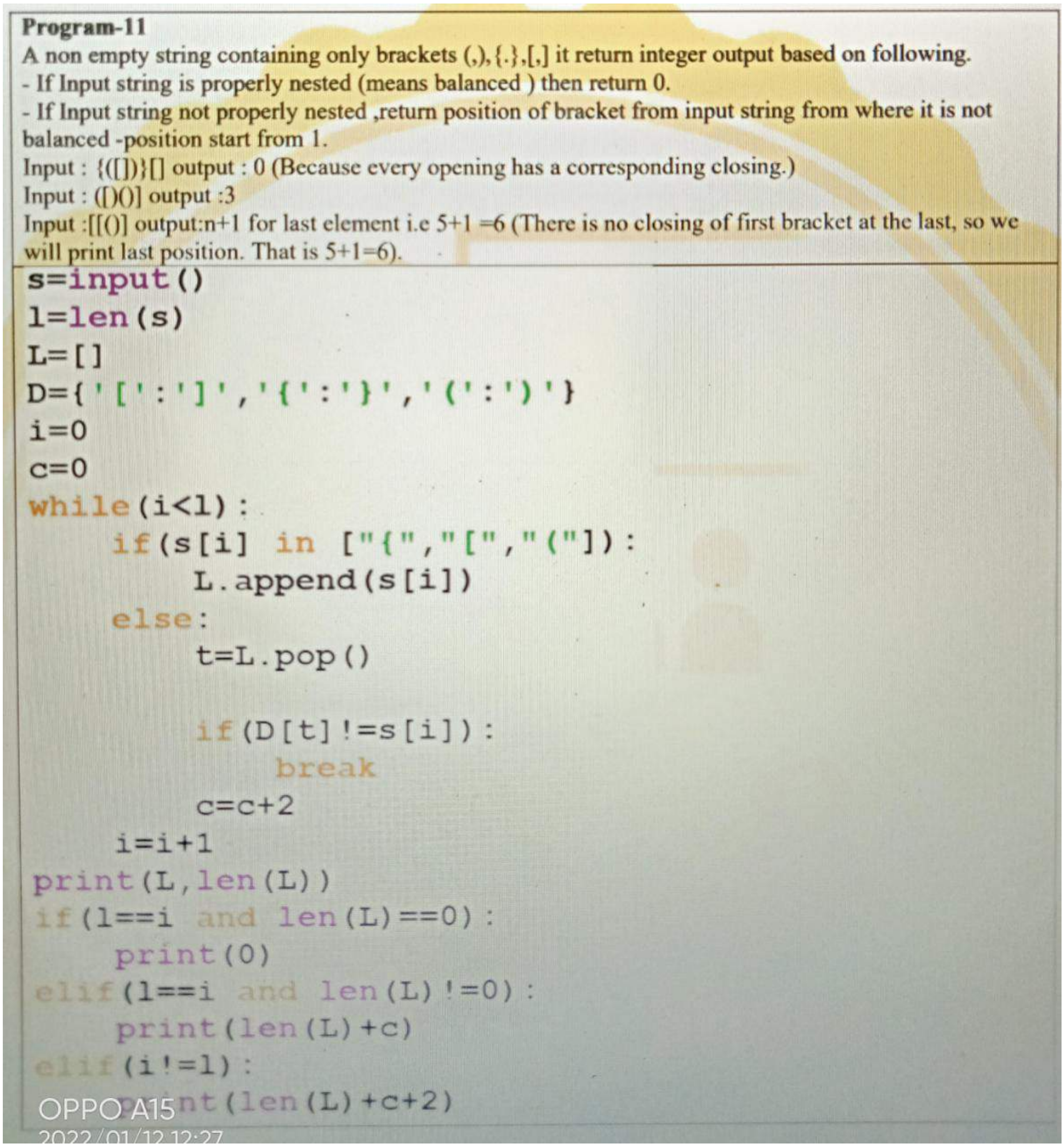


Program-10:

Given input of array of string in format <emp name>: <emp number> separated by comas.
You have to generate password for
Input:- Robert:36787,Tina:68721,Jo:56389
Output :tiX
Find max digit in the emp number which is less or equal to the length of string.And add that place char into the followed by first two letters .If there is no any digit which satisfy the condition then add 'X' into the final string.

```
L=list(input().split(","))
for i in L:
    s=i.split(":")
    l=len(s[0])
    list(s[1]).sort()
    k=-1
    for j in s[1]:
        if(int(j)>l):
            break
        k=int(j)
    #print(k)
    if(k==-1):

        print(s[0][:2]+"X")
    else:
        print(s[0][:2]+s[0][k-1])
```



Program-11

A non empty string containing only brackets (,),{.},[,] it return integer output based on following.

- If Input string is properly nested (means balanced) then return 0.
- If Input string not properly nested ,return position of bracket from input string from where it is not balanced -position start from 1.

Input : {([])}[] output : 0 (Because every opening has a corresponding closing.)

Input : ([)()] output :3

Input :[[()] output:n+1 for last element i.e 5+1 =6 (There is no closing of first bracket at the last, so we will print last position. That is 5+1=6).

```
s=input()
l=len(s)
L=[]
D={'[':']','{':'}','(':')'}
i=0
c=0
while(i<l):
    if(s[i] in ["{","[","("]):
        L.append(s[i])
    else:
        t=L.pop()

        if(D[t]!=s[i]):
            break
        c=c+2
    i=i+1
print(L,len(L))
if(l==i and len(L)==0):
    print(0)
elif(l==i and len(L)!=0):
    print(len(L)+c)
elif(i!=l):
    print(len(L)+c+2)
```

Program-12

A non empty string containing only alphabets . print the longest prefix from input string which is same as suffix.

Prefix and suffix should not be overlapped.

Print -1 if no prefix exits which is also the suffix without overlap.

Do case sensitive comparison.

Position start from 1.

Input :"xxAbcxxAbcxx" o/p : xx ('xx' in the prefix and 'xx' in the suffix and this is longest one in the input string so output will be 'xx').

Input :"Racecar" o/p:-1 (There is no prefix which is in suffix so output will be -1).

```
s=input()
i=1
l=len(s)-1
t=""
while(i<=l):
    if(s[0:i]==s[l:]):
        t=s[0:i]
    i=i+1
    l=l-1
if(t==""):
    print(-1)
else:
    print(s[0:i-1])
```

Program-13

Number of odd sub arrays.

Find the number of distinct subarrays in an array of integers such that the sum of the subarray is an odd integer , two subarrays are considered different if they either start or end at different index.

Input:

```
1
3
1 2 3
```

Output:

```
4
```

Explanation : Total subarrays are [1], [1, 2], [1, 2, 3], [2], [2, 3], [3]

In this there is four subarrays which sum is odd i.e: [1],[1,2] ,[2,3],[3].

```
L=list(map(int,input().split(" ")))
l=len(L)
C=0
for i in range(l):
    for j in range(i+1,l+1):
        if(sum(L[i:j])%2):
            C+=1
print(C)
```

Program-14

An array is given suppose a =[3,5,8,2,19,12,7,11]
One have to find the largest subarray that the element satisfy the following condition x[i]=x[i-1]+x[i-2]
If more than one subarray found then largest one has to be print. And if two subarrays has same number of elements then we will print that subarray which will start from minimum number.
Here the subarrays [2,3,5,8] ,[3,8,11],[5,7,12,19] which are satisfying the above condition.
Expected output is [2,3,5,8].

```
L=list(map(int,input().split()))
L.sort()
l=len(L)
R=[]
i=0
t=[]
while(i<l-1):
    k=0
    a=L[i]
    b=L[i+1]
    while( a+b in L):
        if(len(R)==0):
            R=[a,b,a+b]
        else:
            R.append(a+b)
        c=a+b
        a=b
        b=c


    if(len(t)<len(R)):
        t=R
    R=[]
    i=i+1
print(t)
```

Program-15

A string is given we have to find the longest substring which is unique (that has no repetition) and min size should be 3.

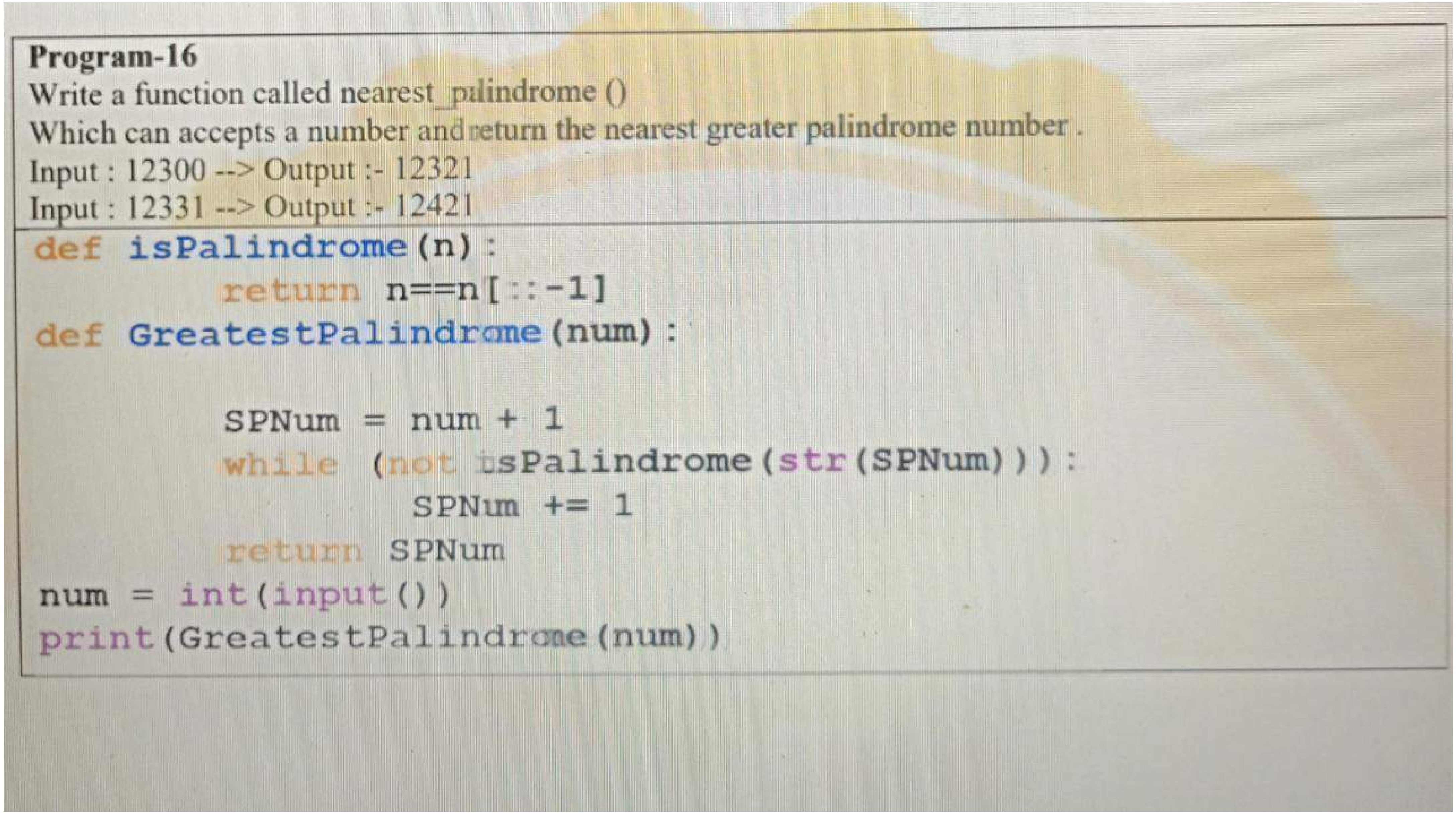
If more than one sub string is found with max length then we have to print one which appeared first in the string.

If no substring is present which matches the condition then we have to print -1;

Ex :input : "A@bcd1abx"

Output : "A@bcd1"

```
s=input()
s=s.lower()
l=len(s)
i=0
R=""
t=""
while(i<l):
      k=0
      while(i+k<l and s[i+k] not in R):
            R+=s[i+k]
            k=k+1
      if(len(t)<len(R)):
            t=R
      i=i+1
      R=""
print(t)
```



Program-16

Write a function called nearest_palindrome ()

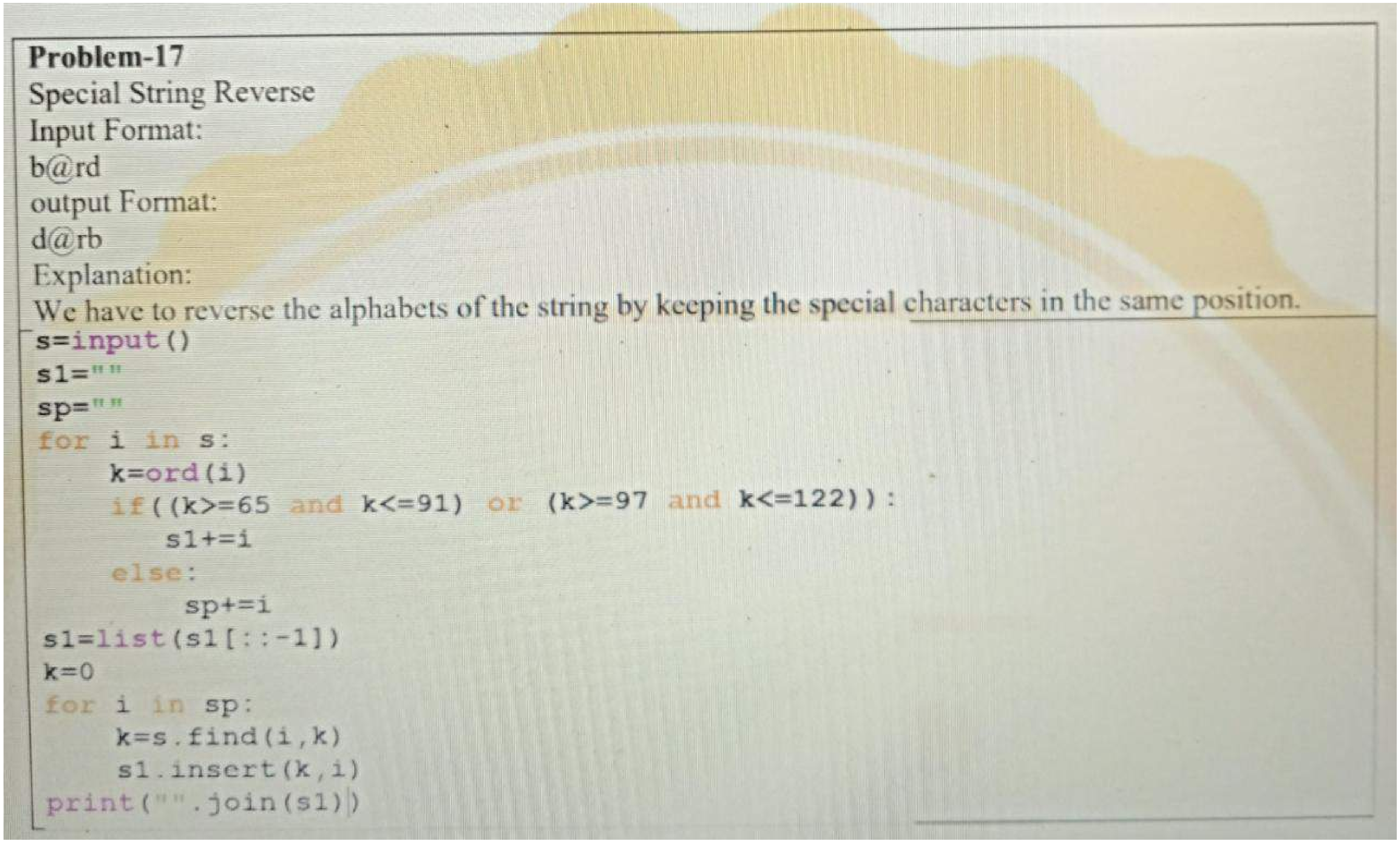
Which can accepts a number and return the nearest greater palindrome number .

Input : 12300 --> Output :- 12321

Input : 12331 --> Output :- 12421

```
def isPalindrome(n):
        return n==n[::-1]
def GreatestPalindrome(num):

        SPNum = num + 1
        while (not isPalindrome(str(SPNum))):
                SPNum += 1
        return SPNum
num = int(input())
print(GreatestPalindrome(num))
```



Problem-17

Special String Reverse

Input Format:

b@rd

output Format:

d@rb

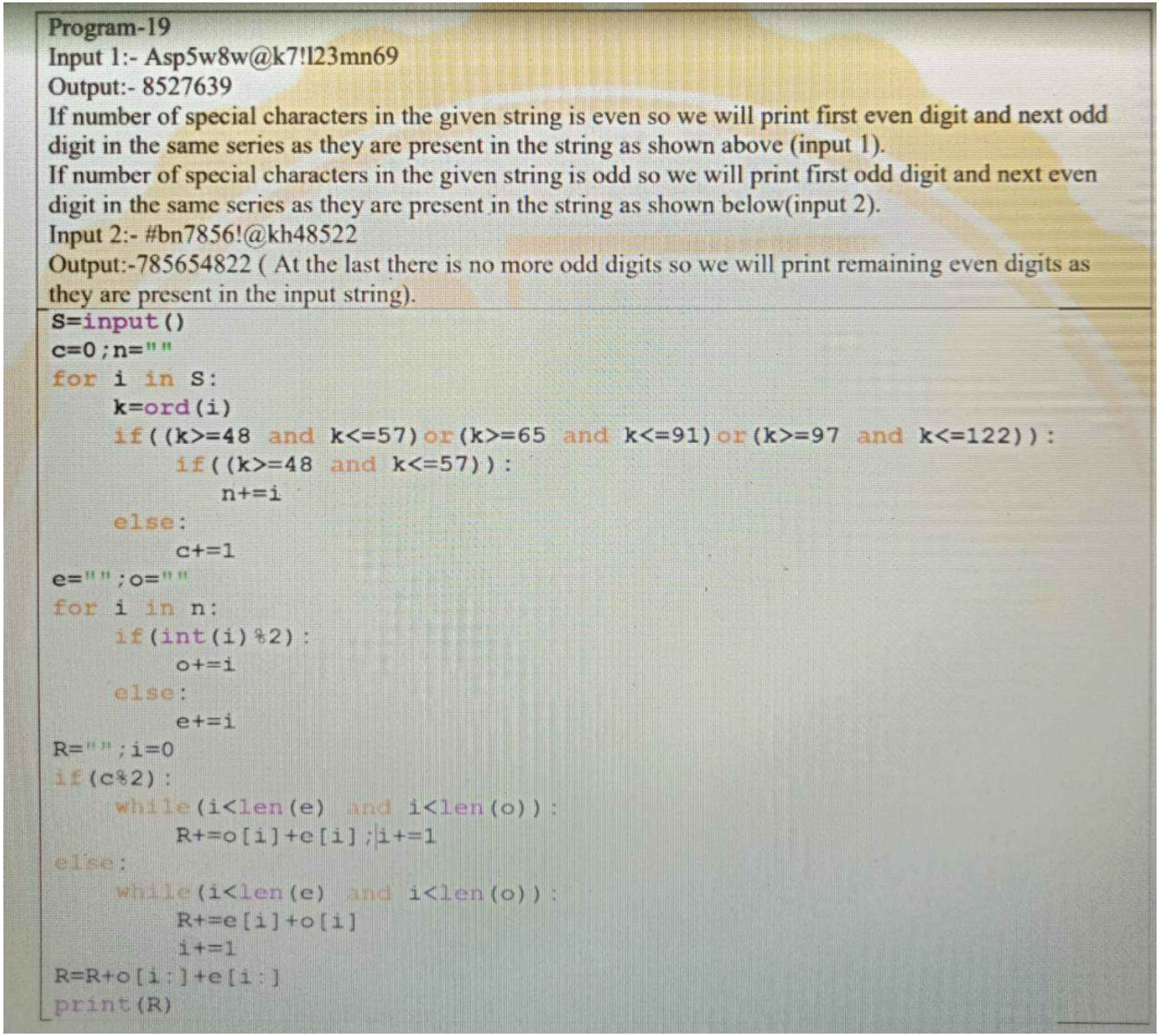
Explanation:

We have to reverse the alphabets of the string by keeping the special characters in the same position.

```
s=input()
s1=""
sp=""
for i in s:
    k=ord(i)
    if((k>=65 and k<=91) or (k>=97 and k<=122)):
        s1+=i
    else:
        sp+=i
s1=list(s1[::-1])
k=0
for i in sp:
    k=s.find(i,k)
    s1.insert(k,i)
print("".join(s1))
```

Program-18
OTP Generation
Input Format: 134567
Output Format:1925
Explanation:
Take the string of numbers and generate a four
digit OTP such that.
1.If the number is odd square it.
2.If the number is even ignore it.
So in the input number 1,3,5,7 are odd.
Square each digits we will get.192549 now print first four digits as output.i.e 1925.
If in case it is not possible then print -1.

```
s=(input())
L=[]
for i in s:
    if(int(i)%2):
        L.append(str(int(i)*int(i)))
s="".join(L)
if(len(s)<4):
    print(-1)
else:
    print(s[:4])
```



Program-19

Input 1:- Asp5w8w@k7!123mn69

Output:- 8527639

If number of special characters in the given string is even so we will print first even digit and next odd digit in the same series as they are present in the string as shown above (input 1).

If number of special characters in the given string is odd so we will print first odd digit and next even digit in the same series as they are present in the string as shown below(input 2).

Input 2:- #bn7856!@kh48522

Output:-785654822 (At the last there is no more odd digits so we will print remaining even digits as they are present in the input string).

```
S=input()
c=0;n=""
for i in S:
    k=ord(i)
    if((k>=48 and k<=57)or(k>=65 and k<=91)or(k>=97 and k<=122)):
        if((k>=48 and k<=57)):
            n+=i
    else:
        c+=1
e="";o=""
for i in n:
    if(int(i)%2):
        o+=i
    else:
        e+=i
R="";i=0
if(c%2):
    while(i<len(e) and i<len(o)):
        R+=o[i]+e[i];i+=1
else:
    while(i<len(e) and i<len(o)):
        R+=e[i]+o[i]
        i+=1
R=R+o[i:]+e[i:]
print(R)
```

Program-20

Find the longest palindrome substring from a string

Input : moomso

Possible cases

Moom , mom , oso , ooo , omo

Longest palindrome substring in the above string is moom so output will be moom.

```
def printSubStr(str, low, high):

        for i in range(low, high + 1):
                print(str[i], end = "")
def longestPalSubstr(str):
        n = len(str)
        maxLength = 1
        start = 0
        for i in range(n):
                for j in range(i, n):
                        flag = 1
                        for k in range(0, ((j - i) // 2) + 1):
                                if (str[i + k] != str[j - k]):
                                        flag = 0
                        if (flag != 0 and (j - i + 1) > maxLength):
                                start = i
                                maxLength = j - i + 1


        printSubStr(str, start, start + maxLength - 1)

longestPalSubstr(input())
```