Ex 1)

Input : a string of comma separated numbers.
The numbers 5 and 8 are present in the list
Assume that 8 always comes after 5.

Case 1: num1 = add all numbers which do not lie between 5 and 8 in the input.

Case 2 : num2= numbers formed by concatenating all numbers from 5 to 8 .
Output: sum of num1 and num2

Example: 1)3,2,6,5,1,4,8,9
Num1:3+2+6+9 =20
Num2:5148
O/p=5248+20 = 5168

Ans :
---

```
# number sequence
ara = list(map(int,input().split(",")))
num1 = sum(ara[:ara.index(5)])+sum(ara[ara.index(8)+1:])
print(num1)
l = ara[ara.index(5):ara.index(8)+1]
#print(l)
num2 = ""
for i in l:
        num2+=str(i)
```

```
#print(num2)
print(int(num2)+num1)
```

---

Ex 2)

A string which is a mixture of letter and integer and special char from which find the largest even number from the available digit after removing the duplicates.

If an even number is not formed then return -1.

Ex : infosys@337
O/p : -1

Hello#81@21349
O/p :983412

Ans:

```
# largest even number
import itertools
s = input()
ss = set()
m=-1
for i in s:
        if i.isdigit():
```

```
            ss.add(i)
ll = list(itertools.permutations(ss,len(ss)))
for i in ll:
        k="".join(i)
        if int(k)%2==0 and int(k)>m:
            m=int(k)
print(m)
```

---

Ex 3)

Write a python program that it should consist of special char, numbers and chars . if there are even numbers of special chars Then 1) the series should start with even followed by odd

Input: t9@a42&516

Output: 492561

If there are odd numbers of special chars then the output will be starting with odd followed by even

Input:5u6@25g7#@

Output:56527

If there are any number of additional digits append them at last

Ans:

---

```
#special chars count
s = input()
c=0
even = []
odd = []
for i in s:
	if i.isalnum():
		c+=1
	if i.isdigit():
		if int(i)%2==0:
			even.append(i)
		else:
			odd.append(i)
print(c)
print(even)
print(odd)
if c%2==0:
	if len(even) > len(odd):
		t = len(odd)
		out = even
	else:
		t=len(even)
		out=odd
	for i in range(t):
		print("{}{}".format(even[i],odd[i]),end="")
	for j in out[t:]:
```

```
            print(j,end="")
else:
        if len(even) > len(odd):
            t = len(odd)
            out = even
        else:
            t=len(even)
            out = odd
        for i in range(t):
            print("{}{}".format(odd[i],even[i]),end="")
        for j in out[t:]:
            print(j,end="")
```

---

Ex 4)

Read ‘m’ m>4
N=m+!
Take m*n matrix
If any num is consecutive for 3 times either in row ,column ,diagonals print the num , if there multiple num print min of those num

Ex: m=6 take 6*7 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

Ans:

---

```
row = int(input())
mat =[]
for i in range(row):
        mat.append(list(map(int,input().split())))
print(mat)
col= len(mat[0])
out=[]
for r in range(row):
        for c in range(col-2):
                if mat[r][c]==mat[r][c+1]==mat[r][c+2]:
                        out.append(mat[r][c])
for r in range(row-2):
        for c in range(col):
                if mat[r][c]==mat[r+1][c]==mat[r+2][c]:
                        out.append(mat[r][c])
for r in range(row-2):
        for  c in range(col-2):
                if mat[r][c]==mat[r+1][c+1]==mat[r+2][c+2]:
                        out.append(mat[r][c])
print(out)
```

```
print(min(out))
```

---

Ex 5)

N and an array where 0<N<len(Array)

Ex : N=2
Array  =1,2,3,3,4,4

O/p :
To find the least number of unique elements after deleting N numbers of elements of numbers in the array

In the above ex , after deleting N=2 number of elements from the array
In above 1,2 should be deleted

3,3,4,4 will be remaining so,
2 unique elements from the array

So ,output in should be 2

Ans:

---

```
import collections
n = int(input())
```

```
ara = list(map(int,input().split(",")))
l = dict(collections.Counter(ara))
l=dict(sorted(l.items(), key=lambda x: x[1]))
#print(l)
ll=[]
for i ,j in l.items():
	#print(i,j)
	if n!=0:
		n-=j
		ll.append(i)
	if n==0:
		break
#print(ll)
c=0
for i in set(ara):
	if i not in ll:
		c+=1
print(c)
```

---

Ex 6)

Maximum number swaps that you can perform on the given array =n

Output:

The final answer should be possible integer that you can get from the given array by performing N swaps

Ans :

---

---

Ex 7)
String rotation
Input rhdt:246,ghftd:1246

Expl :here every string is associated with the number sep by : if sum of squares of digits is even then rotate the string by 1 if square of digits is odd then rotate the string left by 2 position

2*2+4*4+6*6=56 which is even so rotate rhdt --->trhd

1*1+2*2+4*4+6*6=57  which is odd then rotate string by 2 at left “ghftd”  op: ftdgh

Ans:

```
#string rotation
s = input().split(",")
stt=[]
numm=[]
for i in s:
        s1,n = i.split(":")
        stt.append(s1)
        numm.append(n)
def rotate(ss,n):
```

```
    n = list(str(n))
    s =0
    print(n)
    for i in n:
        s+=int(i)**2
    if s%2==0:
        return ss[-1:]+ss[:-1] #right rotation
    else:
        return ss[2:]+ss[:2] #left rotation
for i in range(len(numm)):
    print(rotate(stt[i],numm[i]))
```

---

Ex 8)

Print all 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

Ans:

---
---

Ex 9)
Given input of array of string in format <emp name> <emp number> separated by comas ,
Emp should contain only alphabets and employee number.

You have to generate password for
Ex : input Robert:36787,Tina:68721,Jo:56389

Output :tiX

Conditions: len of robert is 6 and 6 is present in emp number robert (36787),so return the alphabet at position 6 that is t.

Now len of tina is 4 and 3 is not present in the 68721 so select the number which is max and less than the len of tina so select 2 return the alphabet that is at position 2 that is i.

Now In of Jo is 2 it is not present in 56389 and there is not present any number which is less than 2 so return X.

Ans :

```
#password generation
s = input().split(",")
print(s)
stt=[]
numm=[]
for i in s:
	s1,n = i.split(":")
	stt.append(s1)
	numm.append(n)
print(stt)
print(numm)
def pas(ss,n):
	l=len(ss)
	while l!=0:
		if str(l) in n:
			return ss[l-1]
		else:
			l-=1
	return "X"


for i in range(len(numm)):
```

```
        print(pas(stt[i],numm[i]),end="")
```

---

Ex 10)
A non empty string instr containing only parenthesis (,),{.},[,] it return outstr based on following,

- instr is properly nested and return 0
- instr not properly nested ,return position of element in instr
-position start from 1

Input : {([])}[] output : 0
Input : ([)()] output :3
Input :[[()] output:n+1 for last element i.e 5+1 =6

Ans:

---

```
st=[]
ope =['[','{','(']
clo =[']',"}",")"]
def check(s):
    for i in range(len(s)):
        if s[i] in ope:
            st.append(s[i])
        elif s[i] in clo:
```

```
            last = clo.index(s[i])
            #print(last)
            if (len(st) > 0) and (ope[last] == st[len(st) - 1]) :
                st.pop()
            else:
                return ("at position",i+1)
    if len(st) == 0:
        return "good string"
    else:
        return len(s)+1
s = input()
print(check(s))
```

---

Ex 11)

```
import itertools
n1 = int(input())
n2 = int(input())
ara = [i for i in range(n1,n2+1)]
ll = [ara[i:j+1] for i in range(len(ara)-1) for j in range(i,len(ara))]
print(ll)
print(len(ll)-1)
```

A non empty str containing only alphabets . print the longest prefix in str which is same as suffix.

Prefix and suffix should not be overlapped
Print -1 if no prefix exits which is also the suffix without suffix without overlap

Do case sensitive comparison wherever necessary
Position start from 1.

Input :”xxAbcxxAbcxx”  o/p :2
Input :”Racecar”  o/p:-1

Ans :

```
#longest prefix and suffix
s = input()
rev = s[::-1]
print(rev)
c=0
for i in range(len(s)):
	if s[i]==rev[i]:
		c+=1
	else:
		if i==0:
			c=-1
		break
print(c)
```

---

Ex 12)
Get 2 strings as input and find substring for the string from left to right

I/p :  storcp

torcp
Subsequence : s,t,o,top,trp

1> If there is 2 or more largest subsequence then check in string 2 , which subsequence is formed first than print that
2> If there is no any subsequence then print X

Ex : storcp , torap o/p : top
Fryhead , ction o/p: X

Ans :

---
---

Ex 13)
Number of odd sub arrays
Find the number of distinct subarrays in an array of position integers such that the sum of the subarray is an odd integer , two subarray are considered
different if they either start or end at different index.

Input:
1
3
1 2 3

Output:
4

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

Ans :

---

```
import itertools
n1 = int(input())
n2 = int(input())
ara = [i for i in range(n1,n2+1)]
# for i in range(n1,n2+1):
#     ara.append(i)
print(ara)
ll = [ara[i:j+1] for i in range(len(ara)) for j in range(i,len(ara))]
# ll=[]
# for i in range(len(ara)):
#     for j in range(i,len(ara)):
#         ll.append(ara[i:j+1])
print(ll)

c=0
for i in ll:
    if sum(i)%2!=0:
        c+=1
print(c)
```

---

Ex 14)
Find the largest substring

Input : A@B@C1bba
Output:A@B@C1

Substrings are :

A->A@B@C1
@->@b@C1
B->B@C1
@->C1

Since first substring has largest length it will print

Ans:

---
---

Ex 15)
Find the all possible  2*2 matrix whose each should
Follow rule that each element of 2*2 matrix should be divisible by sum of its digits.

Ex :
N=4
42 54 2

30 24 27
180 190 40
11 121 13

O/p:
42 54
30 24
30 24
180 190
24 27
190 40

Ans :

---

```
def harshad(n):
	s = sum(list(map(int,str(n))))
	if n%s==0:
		return True
	else:
		return False
row = int(input())
mat = []
for i in range(row):
	mat.append(list(map(int,input().split())))
col=len(mat[0])
for i in range(row-1):
	for j in range(col-1):
```

```
            if harshad(mat[i][j]) and harshad(mat[i][j+1]) and
harshad(mat[i+1][j]) and harshad(mat[i+1][j+1]):
                print("{},{}".format(mat[i][j],mat[i][j+1]))
                print("{},{}".format(mat[i+1][j],mat[i+1][j+1]))
```

---

Ex 16) max subarray

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 substring if found then largets one has to print the array which starts with the minimum elements and if they are also same then the array with minimum second element and so on .

Here the subarrays [2,3,5,8] ,[3,8,11],[5,7,12,19]
Expected is [2,3,5,8]

Ans:

---

```
#max subarray from list
aa = list(map(int,input().split()))
aa=sorted(aa)
j = []
```

```
j.append(aa[0])
j.append(aa[1])
print(j)
for i in range(2,len(aa)):
        if j[i-1]+j[i-2] in aa:
                j.append(j[i-1]+j[i-2])
        else:
                break
print(j)
```

---

Ex 16)

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

If more than one sub string is found with max length the we have to print one which appered first in thw string
If no substring is present which matches the condition then we have to print -1;

Ex :input : “A@bcd1abx”
Output : “A@bcd1”

Ans :

---

```
# unique substring
s = input()
b=""
for i in range(len(s)):
	if s[i].lower() in b or s[i].upper() in b:
		break
	else:
		b+=s[i]
print(b)
```

---

Ex 17)
For a given list of numbers find the its factors and add the factors then if the sum of all factor is present in original list , sort it and print it

Ex :
Input: 0,1,6
Factors 0 = 0 , sum =0
1=1 sum =1
6 =1,2,3 = sum =6

Output : 1,6
If the sum is not present in the list then return -1.

Ans:

---

```
#factor addition
def factor(n):
        s=0
        for i in range(1,n):
                if n%i==0:
                        s+=i
                        print(i,s)
        print(s)
        return s

list1 = list(map(int,input().split()))
for i in list1:
        if factor(i) in list1:
                print(i)
```

---

Ex 18)

Write a python function nearest_palindrome ()

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

Input : 123000  --> 12321

Input : 12331  --> 12421

Ans :

---

```
n = int(input())
while True:
        rev = int(str(n)[::-1])
        if rev==n:
                print(int(n))
                break
        n+=1
```

---

Ex 19)

1:special  string reverse

Input Format:

b@rd

output Format:

d@rb

Explanation:

We should reverse the alphabets of  the  string by keeping the special characters in  the  same  position

Ans :

---

```
s = input()
d = dict()
rev=""
for i in range(len(s)):
    if s[i].isalnum()==False:
        d.update({i:s[i]})
    else:
        rev+=s[i]
```

```
print(d)
rev = list(rev[::-1])
for i ,j in d.items():
    rev.insert(i,j)
print("".join(rev))
```

---

Ex 20)

OTP Generation

Input Format: 13456

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.

Ans:

---

```
n=input()
L=['0','1','4','9','16','25','36','49','64','81','100']
s=""
for i in n:
    if int(i)%2==0:
        continue
    else:
        s+=L[int(i)]
print(s[:4])
```

---

Ex 21)
input:- Asp5w8w@k7!l23mn69

Output:- 8527639

As num of spl characters in the given string is even so we should print first even digits and
next odd digits in the same series present in the string

Input:-
#bn7856!@kn2n65jbnj482375
Output:-
7856523674582

As count of spl characters in the given string is odd so we should first print odd digits and
then even digits in the same series present in the string

Ans :

---
```
#special chars count
s = input()
c=0
even = []
```

```
odd = []
for i in s:
	if i.isalnum()==False:
		c+=1
	if i.isdigit():
		if int(i)%2==0:
			even.append(i)
		else:
			odd.append(i)
print(c)
print(even)
print(odd)
if c%2==0:
	if len(even) > len(odd):
		t = len(odd)
		out = even
	else:
		t=len(even)
		out=odd
	for i in range(t):
		print("{}{}".format(even[i],odd[i]),end="")
	for j in out[t:]:
		print(j,end="")
else:
	if len(even) > len(odd):
		t = len(odd)
		out = even
	else:
		t=len(even)
```

```
            out = odd
        for i in range(t):
            print("{}{}".format(odd[i],even[i]),end="")
        for j in out[t:]:
            print(j,end="")
```

---

Ex 22)
Input:- 93012630
Output:- 2,6,12,30,930,

We should divide the total number into substrings and we should verify each num is pronic num or
not if pronic we should print that num
Pronic: means it is a multiple of two consecutive integers

Ex: 6->2*3 it’s a pronic
12->3*4 it’s a pronic

Input: 12665042
Output:- 2,6,12,42,650

Ans :

---

```
# pronic number
def pronic(n):
```

```
        for i in range(1,(n//2)+1):
                if i*(i+1)==n:
                        return True
        return False
x = input()
ara = [x[i:j+1]for i in range(len(x)) for j in range(i,len(x))]
print(ara)
final =set()
for i in ara:
        if pronic(int(i)):
                final.add(int(i))
print(sorted(final))
```

---

Ex23)

*Parking slots-*
A, B, C, D - 4 lanes with each capacity 10 slots

*1* First take input for the 4 lanes with booked slots 1-10 for each lane (4 inputs in 4 lines and
values separated by commas)
*2* If no slot is booked for a lane take '-1' as input for that lane
*3* take input for waiting cars ⬜⬜⬜⬜⬜⬜⬜⬜⬜
*To perform*

*----------*

*4* for all waiting cars to be parked
first find out the lane with maximum free slots and

if 2 slots have same no of highest free slots then prefer A-D flow

*5* Fill each waiting car, in the selected lane and mark it as the seq no for the lane with free
slots A1, A2.....A10 till all waiting cars are parked
If A has already filled 5 slots then fill waiting slots from A6-A10

*6* If the waitit cars ********** are not completely parked even after the highest free lane was
completely filled, then fill the next highest free lane
Continue till all waiting cars are parked

*Output*
*----------*

Print all the booked slot series of waiting cars
Like-A7 A8 A9 A10 C10 D10

*Note*:
*---------*
If no slot is fee and cars ********** can't be parked then print capital 'X'
(Just a test case with 6% for this last constraint)

Ans :

---

---

Ex24)

Set of number given and the sum is given

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

Sum=0

Output should be combination of which satisfy the case

(-1,1,2,-2)(0,0,1,-1)(0,0,-2,2)

Output : 3

Ans :

---

```
import itertools
ara = list(map(int,input().split(",")))
s = int(input())
l=list(itertools.combinations(ara,4))
print(l)
c=0
for i in l:
        a = sum(i)
        if a == s:
                c+=1
print(c)
```

---

Ex25)

Find the longest palindrome from a string

Input : moomso

Possible cases
Moom , mom , oso , ooo , omo

Longest is moom so output :moom

Ans:
---
```
s = input()
ara = [s[i:j+1] for i in range(len(s)) for j in range(i,len(s))]
print(ara)
l=0
out=""
for i in ara:
    rev= i[::-1]
    if i == rev and len(i)>l:
        l=len(i)
        out=i
print(out)
```
---

Ex26)
Input :HelLoWOrld
Output: dWerHoOlLl

Instruction:
First get the similar char in combinations like :
['d', 'e', 'H', 'lLl', 'oO', 'r', 'W']

Then concatenate first element and last element wise versa.

dWerHoOlLl

---

```
#======
k = input()
ss = sorted(set(k.upper()))
print(ss)
fin = []
for i in range(len(ss)):
	s=""
	for j in k:
		if ss[i]==j.upper():
			s+=j
	fin.append(s)
print(fin)
i=0
j=len(fin)-1
while i<=j:
	if i==j:
		out+=fin[i]
	else:
		out+=fin[i]+fin[j]
	i+=1
	j-=1
print(out)
```

---

Ex27)