1) Write a Python program to find the largest and smallest number in an unsorted array.

arr=[int(x) for x in input().split()]

n=len(arr)

lar=small=arr[0]

for i in range(n):

if lar<arr[i]:

lar=arr[i]

for i in range(n):

if (small>arr[i]):

small=arr[i]

print("Largest value =",lar)

print("Smallest value =",small)

2) Write a Python program to implement Insertion Sort.

def insertion\_sort(alist):

for i in range(1, len(alist)):

temp = alist[i]

j = i - 1

while (j >= 0 and temp < alist[j]):

alist[j + 1] = alist[j]

j = j - 1

alist[j + 1] = temp

alist = input().split()

alist = [int(x) for x in alist]

insertion\_sort(alist)

print(alist)

3) Write a Python program to implement Quick Sort

def partition(arr, low, high):

i = (low-1)

pivot = arr[high]

for j in range(low, high):

if arr[j] <= pivot:

i = i+1

arr[i], arr[j] = arr[j], arr[i]

arr[i+1], arr[high] = arr[high], arr[i+1]

return (i+1)

def quickSort(arr, low, high):

if len(arr) == 1:

return arr

if low < high:

pi = partition(arr, low, high)

quickSort(arr, low, pi-1)

quickSort(arr, pi+1, high)

arr = [int(x) for x in input().split()]

n = len(arr)

quickSort(arr, 0, n-1)

print(\*arr,sep = " ")

4) Write a Python program to implement Merge Sort

def mergeSort(arr):

if len(arr)>1:

m=len(arr)//2

L=arr[:m]

R=arr[m:]

mergeSort(L)

mergeSort(R)

i=j=k=0

while i<len(L) and j<len(R):

if L[i]<R[j]:

arr[k]=L[i]

i+=1

else:

arr[k]=R[j]

j+=1

k+=1

while i<len(L):

arr[k]=L[i]

i+=1

k+=1

while j<len(R):

arr[k]=R[j]

j+=1

k+=1

arr=[int(x) for x in input().split()]

print("Given array is")

for i in range(len(arr)):

print(arr[i],end=" ")

print("")

mergeSort(arr)

print("Sorted array is:")

for i in range(len(arr)):

print(arr[i],end=" ")

## 5) SINGLY LINKED LIST

class node :

def \_init\_(self,data) :

self.data = data

self.next = None

class linked\_\_list :

def \_init\_(self) :

self.head =None

def print\_\_list(self) :

temp=self.head

while(temp != None ):

print(temp.data)

temp=temp.next

list = linked\_\_list()

list.head = node(1)

l2 = node(2)

l3 = node(3)

list.head.next = l2

l2.next = l3

list.print\_\_list()

6) double linked list

class Node:

def \_init\_(self,data):

self.data = data;

self.previous = None;

self.next = None;

class InsertStart:

#Represent the head and tail of the doubly linked list

def \_init\_(self):

self.head = None;

self.tail = None;

#addAtStart() will add a node to the starting of the list

def addAtStart(self, data):

#Create a new node

newNode = Node(data);

#If list is empty

if(self.head == None):

#Both head and tail will point to newNode

self.head = self.tail = newNode;

#head's previous will point to None

self.head.previous = None;

#tail's next will point to None, as it is the last node of the list

self.tail.next = None;

#Add newNode as new head of the list

else:

#head's previous node will be newNode

self.head.previous = newNode;

#newNode's next node will be head

newNode.next = self.head;

#newNode's previous will point to None

newNode.previous = None;

#newNode will become new head

self.head = newNode;

#display() will print out the nodes of the list

def display(self):

#Node current will point to head

current = self.head;

if(self.head == None):

print("List is empty");

return;

print("Adding a node to the start of the list: ");

while(current != None):

#Prints each node by incrementing pointer.

print(current.data),

current = current.next;

print();

dList = InsertStart();

#Adding 1 to the list

dList.addAtStart(1);

dList.display();

#Adding 2 to the list

dList.addAtStart(2);

dList.display();

#Adding 3 to the list

dList.addAtStart(3);

dList.display();

#Adding 4 to the list

dList.addAtStart(4);

dList.display();

#Adding 5 to the list

dList.addAtStart(5);

dList.display();

7) **Write a Python program to delete particular element from the list**

arr=[int (x) for x in input().split()]

n=int(input())

arr. remove(n)

print(arr)

8) Write a Python program for implementation of Bubble Sort

def bubble\_sort(arr,n):

for i in range(0,n-1):

flag = 0

for j in range(0,n-i-1):

if(arr[j]>arr[j+1]):

temp = arr[j]

arr[j] = arr[j+1]

arr[j+1] = temp

flag = 1

if(flag == 0):

break

arr=[int(x) for x in input().split()]

n = len(arr)

bubble\_sort(arr,n)

print(arr)

9)FOBONACCI SERIES

a=int(input())

flag=0

for i in range(2,a):

if(a%i==0):

flag+=1

if(flag==0):

print(a,'is a prime number ')

else:

print(a,'is not a prime number')

10) Write a python program to check whether a given number is prime number or not

num=int(input())

if num>1:

for i in range(2,num):

if(num%i)==0:

print(num,"is not a prime number")

break

else:

print(num,"is a prime number")

11) Write a python program to find factorial of a given number

n=int(input())

def calc(n):

fact=1

if n==-5:

return"Factorial does not exist for negative numbers"

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

fact\*=i;

return fact

print(calc(n))

12)check Palindrome

n=int(input(""))

temp=n

rev=0

while(n>0):

dig=n%10

rev=rev\*10+dig

n=n//10

if(temp==rev):

print("Palindrome Number")

else:

print("Not a Palindrome number")

13) Leap Year or not

year = int(input())

if (( year%400 == 0)or (( year%4 == 0 ) and ( year%100 != 0))):

print("Leap Year")

else:

print("Not a Leap Year")

14) Write a Python Program to Display Floyd's Triangle

rows = int(input(""))

number = 1

print("Floyd's Triangle")

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

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

print(number, end = ' ')

number = number + 1

print()

15) Find the Highest Common Factor of two numbers

n,m=int(input()),int(input())

min=min(n,m)

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

if n%i==0 and m%i==0:

hcf=i

print("The H.C.F. is",hcf)

16)MATRIX ADDITION

X=[[1,2,3],

[4,5,6],

[7,8,9]]

Y=[[10,11,12],

[13,14,15],

[16,17,18]]

sum=[[0,0,0],

[0,0,0],

[0,0,0]]

for i in range(len(X)):

for j in range(len(X[0])):

sum[i][j]=X[i][j]+Y[i][j]

for num in sum:

print(num)

17) matrix multiplication

X = [[12,7,3],

[4 ,5,6],

[7 ,8,9]]

Y = [[5,8,1,2],

[6,7,3,0],

[4,5,9,1]]

result =[[0,0,0,0],

[0,0,0,0],

[0,0,0,0]]

for i in range(len(X)):

for j in range(len(Y[0])):

for k in range (len(Y)):

result[i][j]+=X[i][k]\*Y[k][j]

for r in result:

print(r)

18) Write a Python program to print all Happy Numbers between 1 and 100

def squaresum(n):

s=0

while n!=0:

r=n%10

n=n//10

s+=r\*\*2

return s

for i in range(1,101):

s=i

while s!=1 and s!=4:

s=squaresum(s)

if s==1:

print(i, end=" ")

19) Write a Python program to check whether the given character is vowel or consonant.

l1=input()

if(l1=='a' or l1=='e' or l1=='i' or l1=='o' or l1=="u"):

print (f'{l1} is a Vowel')

else:

print (f'{l1} is a Consonant')

# 20)C Program to check whether the given number is Armstrong Number or not.

#include<stdio.h>

int main()

{

int n,temp,r,sum=0;

scanf("%d",&n);

temp = n;

while(temp!=0)

{

r=temp%10;

sum = sum+r\*r\*r;

temp/=10;

}

if(sum == n)

printf("armstrong number");

else

{

printf("armstrong number");

}

}

21) to remove the punctuations in a string

test\_str = input()

punctuations = '''!@#$%^&\*([{)]}-:;"\'<.,>/?~.\_'''

for ele in test\_str:

if ele in punctuations:

test\_str = test\_str.replace(ele, "")

print(test\_str)

22) Write a Python program to perform Decimal to Binary Conversion using recursive function

def Binary(n):

if n>1:

Binary(n//2)

print(n % 2,end=' ')

dec = int(input())

Binary(dec)

print()

23) Write a Python program to determine whether the given number is a Harshad Number or not

num = int(input())

sum = 0

temp = num

while temp:

rem = temp%10

sum += rem

temp//=10

if num%sum == 0:

print(num ,"is a harshad number")

else:

print(num ,"is not a harshad number")

24) Write a Python program to find the largest odd and event number in the array

def count(array,n):

a=[False for i in range(n)]

for i in range(n):

if(a[i]==True):

continue

count=1

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

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

a[j]=True

count+=1

print(array[i],'',count)

array=[int(x) for x in input().split()]

n=len(array)

count(array,n)

25) ) Write a C program to find the search element in an array using Linear Search

#include <stdio.h>

int main()

{

int array[100], search, c, n;

scanf("%d", &n);

for (c = 0; c < n; c++)

scanf("%d", &array[c]);

scanf("%d", &search);

for (c = 0; c < n; c++)

{

if (array[c] == search) /\* If required element is found \*/

{

printf("%d \n", c+1);

break;

}

}

if (c == n)

printf("%d isn't present in the array.\n", search);

return 0;

}

26) Write a C program to find the average of n elements in an array

#include<stdio.h>

int main()

{

int array[100],n,i,sum=0,avg;

scanf("%d",&n);

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

scanf("%d",&array[i]);

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

{

sum=sum+array[i];

}

avg=sum/n;

printf("%d\n",avg);

}

27) Write a Python program to implement queue using linked list

class Node:

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

self.data=data

self.next=None

class Queue:

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

self.front=self.rear=None

def isEmpty(self):

return self.front==None

def EnQueue(self, item):

temp=Node(item)

if self.rear==None:

self.front=self.rear=temp

return

self.rear.next=temp

self.rear=temp

def DeQueue(self):

if self.isEmpty():

return

temp= self.front

self.front=temp.next

if(self.front == None):

self.rear=None

if \_\_name\_\_== '\_\_main\_\_':

q = Queue()

print("IMPLEMENTATION OF QUEUE USING LINKED LIST")

print("\n Enqueue 10, 20")

q.EnQueue(10)

q.EnQueue(20)

print("\n Dequeue Twice")

q.DeQueue()

q.DeQueue()

print("\n Enqueue30, 40, 50")

q.EnQueue(30)

q.EnQueue(40)

q.EnQueue(50)

print("\n Dequeue Once")

q.DeQueue()

print("\n Queue Front "+ str(q.front.data))

print("\n Queue Rear "+ str(q.rear.data))

28) Write a Python program to implement Queue using Arrays

class Queue:

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

self.queue=[]

self.front=self.rear=0

self.capacity=c

def queueEnque(self,data):

if(self.capacity==self.rear):

print("\nQueue is full")

else:

self.queue.append(data)

self.rear+=1

def queueDeque(self):

if(self.front==self.rear):

print("\nQueue is Empty")

else:

x=self.queue.pop(0)

self.rear-=1

def queueDisplay(self):

if(self.front==self.rear):

print("\nQueue is Empty")

for i in self.queue:

print(i,"<--",end='')

def queueFront(self):

if(self.front==self.rear):

print("\nQueue is Empty")

print("\nFront Element is:",self.queue[self.front])

if \_\_name\_\_=='\_\_main\_\_':

q=Queue(4)

print("SIZE OF THE QUEUE = 4")

q.queueDisplay()

print("After Insertion of 10, 20, 30, 40")

q.queueEnque(10)

q.queueEnque(20)

q.queueEnque(30)

q.queueEnque(40)

q.queueDisplay()

q.queueEnque(50)

q.queueDisplay()

q.queueDeque()

q.queueDeque()

print("\n\nafter two node deletion\n")

q.queueDisplay()

q.queueFront()

29) Write a Python program to implement stack using linked list

print("IMPLEMENTATION OF STACK USING LINKED LIST")

stack = []

stack.append(10)

stack.append(20)

stack.append(30)

print(" 10 pushed to stack")

print(" 20 pushed to stack")

print(" 30 pushed to stack")

print("",stack.pop(), "popped from stack")

print("Top element is 20")

30) Write a Python program to implement stack using arrays

print("IMPLEMENTATION OF STACK USING ARRAYS")

stack=[]

stack.append(10)

stack.append(20)

stack.append(30)

print("10 pushed to stack")

print("20 pushed to stack")

print("30 pushed to stack")

print(stack.pop(),"popped from stack")