**Day-1**

Intro

Test

Problems on Dictionaries

**Day-2**

Functions in Python – Theory

Done in online gdb

Import modules

We can use a function return in one file in another file by importing first file-------need to save first file with .py extension

**Factorial of list of numbers**

def fact(n):

if n<=1:

return 1

else:

return n\*fact(n-1)

f=[1,2,3,4,5]

re=[]

for i in range(len(f)):

s=fact(f[i])

re.append(s)

print(re)

variables inside the functions are local.

To make local variable into global we use **global** keyword

**Reverse and palindrome of string using recursion**

def sumre(s):

if len(s)==0:

return s

else:

return sumre(s[1: ])+s[0]

n=input()

s=sumre(n)

if s==n:

print("Yes")

else:

print("No")

**GCD without using recursion**

def gcd(n,m):

mi=min(n,m)

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

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

return i

n=int(input())

m=int(input())

print(gcd(n,m))

**GCD using recursion**

def gcd(n,m):

s=min(n,m)

d=max(n,m)

li=[]

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

if s%i==0:

li.append(i)

if d%s==0:

return s

else:

for i in range(len(s),0,-1):

s=li[i-1]

gcd(s,m)

n,m=map(int,input().split())

print(gcd(n,m))

**or**

def gcd(n,m,i):

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

return i

return gcd(n,m,i-1)

n,m=map(int,input().split())

if n>m:

n,m=m,n

print(gcd(n,m,n))

**prime number using recursion**

def prim(n,m):

if m==1:

return True

elif n%m==0:

return False

else:

return prim(n,m-1)

n=int(input())

print(prim(n,n-1))

**LCM**

def lcm(n,m,gre):

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

return gre

else:

gre+=1

return lcm(n,m,gre)

n=int(input())

m=int(input())

gre=max(n,m)

print(lcm(n,m,gre))

**Fibnocqi without recursion**

n=int(input())

f=0

s=1

print(f)

print(s)

for i in range(n-2):

t=f+s

print(t)

f=s

s=t

**Fibnocqi with recursion**

def fib(a,b,c,n,i=1):

if i>n:

return ""

else:

print(a)

c=a+b

a=b

b=c

fib(a,b,c,n,i+1)

n=int(input())

a=0

b=1

c=a+b

fib(a,b,c,n)

**Day-3**

Data structures Algorithms on searching

**Linear search without recursion**

def linear(a,t):

for i in range(len(a)):

if a[i]==t:

return True

return False

s=list(map(int,input().split()))

k=int(input())

print(linear(s,k))

**Linear search using recursion**

def linear(a,t,i=0):

if i==len(a):

return False

if a[i]==t:

return True

else:

return linear(a,t,i+1)

a=list(map(int,input().split()))

t=int(input())

print(linear(a,t))

**Binary search**

def binary(a,t):

s=0

f=len(a)-1

mid=(s+f)//2

while True:

if a[mid]==t:

return True

elif a[mid]>t:

f=mid-1

else:

s=mid+1

mid=(s+f)//2

if s>f or f<s:

return False

n=list(map(int,input().split()))

h=int(input())

print(binary(n,h))

**Binary search using recurssion**

def binary(a,t,s,f):

mid=(s+f)//2

if s>f:

return False

if a[mid]==t:

return True

elif a[mid]>t:

f=mid-1

return binary(a,t,s,f)

else:

s=mid+1

return binary(a,t,s,f)

a=list(map(int,input().split()))

h=int(input())

g=len(a)-1

print(binary(a,h,0,g))

**Codechef Test**

**C++ intro by DSP sir**

**Day-4**

**Leet Code-2529**

**Maximum count of positive and negative integers**

**Using Binary Search**

def maximumCount(self, nums: List[int]) -> int:

        s=0

        e=len(nums)-1

        p=0

        while True:

            m=(s+e)//2

            if nums[m]>=0 and nums[m-1]<0:

                p=m

                break

            if nums[m]<0:

                s=m

            if nums[m]>0:

                e=m

            if min(nums)>=0:

                return len(nums)-nums.count(0)

            if max(nums)<=0:

                return len(nums)-nums.count(0)

        return max(p,len(nums)-p-nums.count(0))

**Leet Code-374 Guess number higher or lower**

def guessNumber(self, n: int) -> int:

        s=1

        e=n

        while s<=e:

            m=(s+e)//2

            p=guess(m)

            if p==0:

                return m

            elif p==-1:

                e=m-1

            else:

                s=m+1

**Leet Code-2824**

def countPairs(self, nums: List[int], target: int) -> int:

c=0

for i in range(len(nums)):

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

if nums[i]+nums[j]<target:

c+=1

return c

**Leet Code-278**

 def firstBadVersion(self, n: int) -> int:

        s=1

        e=n

        while s<=e:

            m=(s+e)//2

            p=isBadVersion(m)

            s=isBadVersion(m-1)

            if p==True and s==False:

                return m

            elif p==False:

                s=m+1

            else:

                e=m-1

**c++ classes By DSP sir**

**Bitwise operator**

**Program to convert int to binary with bitwise operator**

**def fun(n):**

**j=""**

**while n!=0:**

**s=n&1**

**j+=str(s)**

**n=n>>1**

**return j[::-1]**

**n=int(input())**

**print(fun(n))**

**Day-5**

**Bitwise operations**

**Multiplication Without using \***

n=int(input())

k=int(input())

for i in range(k):

if 2\*\*i==k:

s=i

break

print(n<<s)

**Division without using /**

n=int(input())

k=int(input())

for i in range(k):

if 2\*\*i==k:

s=i

break

print(n>>s)

**Addition without using +**

x=int(input())

y=int(input())

while y!=0:

c=x&y

x=x^y

y=c<<1

print(x)

**Day-6**

**Sub Arrays**

**Total number of subarrays=((n\*(n+1))//2)**

**Program to find all the subarrays :-**

def fun(l):

for i in range(len(l)):

for j in range(i,len(l)):

print(l[i:j+1])

fun([1,2,5,6,7])

Time complexity:-------o(n^3)

Space complexity:--------o(n(n+1)//2)

**Program to find all the subarrays :-**

def fun(l):

li=[]

for i in range(len(l)):

for j in range(i,len(l)):

li.append(l[j])

print(li)

li=[]

fun([1,2,3,4])

**Time complexity:----o(n^2)**