Part A



Part B

Merge tools

def merge\_the\_tools(string, k):

    s=set()

    result=""

    if(k==1):

         for i in string:

            print(i)

         return

    result=string[0]

    s.add(string[0])

    for i in range(1,len(string)):

        if((i+1)%k==0):

            if(string[i] not in s):

                result+=string[i]

            print(result)

            result=""

            s=set()

        else:

            if(string[i] not in s):

                result+=string[i]

                s.add(string[i])

if \_\_name\_\_ == '\_\_main\_\_':

    string, k = input(), int(input())

    merge\_the\_tools(string, k)

2. **Collection Counter**

n=int(input())

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

d={}

for i in l:

    if(i not in d):

        d[i]=1

    else:

        d[i]+=1

customer=int(input())

tot\_sum=0

for i in range(customer):

    c=list(map(int,input().split()))

    if(c[0] in d and d[c[0]]>0):

        tot\_sum+=c[1]

        d[c[0]]-=1

print(tot\_sum)

3.Company Logo

#!/bin/python3

import math

import os

import random

import re

import sys

if \_\_name\_\_ == '\_\_main\_\_':

    s = input()

    d={}

    for i in s:

        if(i not in d):

            d[i]=1

        else:

            d[i]+=1

    sorted\_dict = dict(sorted(d.items(), key=lambda item: (-item[1], item[0])))

    ct=0

    for i in sorted\_dict:

        print(i,sorted\_dict[i])

        ct+=1

        if(ct==3):

            break

4.IfElse:

#!/bin/python

import math

import os

import random

import re

import sys

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(raw\_input().strip())

    if(n%2!=0 ):

        print("Weird")

    elif(n%2==0):

        if(n in range(2,6)):

            print("Not Weird")

        elif(n in range(6,21)):

            print("Weird")

        elif(n>20):

            print("Not Weird")

5.Arithmatic operation

if \_\_name\_\_ == '\_\_main\_\_':

    a = int(input())

    b = int(input())

    print(a+b)

    print(a-b)

    print(a\*b)

6.Python Division

if \_\_name\_\_ == '\_\_main\_\_':

    a = int(input())

    b = int(input())

    print(a//b)

    print(a/b)

7.Loop

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    for i in range(n):

        print((i)\*\*2)

8.Leap Year

def is\_leap(year):

    leap = False

    if (year % 4 == 0):

        if (year % 100 == 0):

            if (year % 400 == 0):

                leap= True

            else:

                leap= False

        else:

            leap= True

    else:

        leap= False

    # Write your logic here

    return leap

year = int(input())

print(is\_leap(year))

9.Print Function

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    for i in range(n):

        print(i+1,end="")

10 List comprehension

if \_\_name\_\_ == '\_\_main\_\_':

    x = int(input())

    y = int(input())

    z = int(input())

    n = int(input())

    l=[]

    for i in range(0,x+1):

        for j in range(0,y+1):

            for k in range(0,z+1):

                if(i+j+k != n):

                    l.append([i,j,k])

    print(l)

Part C

1.Longest substring without repeat character

class Solution:

    def lengthOfLongestSubstring(self, s: str) -> int:

        check=set()

        ct=0

        st=0

        m=0

        for i in range(0,len(s)):

            if(s[i] not in check):

                ct+=1

                check.add(s[i])

            else:

                while(s[st]!=s[i]):

                    check.remove(s[st])

                    ct-=1

                    st+=1

                st+=1

            if(ct>m):

                m=ct

        return m

2.Zig zag conversion

class Solution:

    def convert(self, s: str, n: int) -> str:

        l=[]

        if(n==1):

            return s

        for i in range(n):

            l.append([])

        k=0

        c=0

        for i in s:

            if(c==0):

                l[k].append(i)

                k+=1

            if(c==1):

                l[k].append(i)

                k-=1

            if(k==n):

                c=1

                k-=2

            if(k==-1):

                c=0

                k+=2

            print(k)

        x=""

        for i in l:

            x+="".join(i)

        return x

3. 3sum Closest

class Solution:

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

        closet = float('inf')

        nums.sort()

        for i in range(len(nums) - 2):

            l, r = i + 1, len(nums) - 1

            while l < r:

                sum3 = nums[i] + nums[l] + nums[r]

                print(sum3)

                if sum3 < target:

                    l += 1

                else:

                    r -=1

                if abs(sum3 - target) < abs(closet - target):

                    closet = sum3

        return closet

4.paranthesis

class Solution:

    def generateParenthesis(self, n: int) -> List[str]:

        def dfs(left, right, s):

            if len(s) == n \* 2:

                res.append(s)

                return

            if left < n:

                dfs(left + 1, right, s + '(')

            if right < left:

                dfs(left, right + 1, s + ')')

        res = []

        dfs(0, 0, '')

        return res

5.String Multiplication

class Solution:

    def multiply(self, num1: str, num2: str) -> str:

        return str(int(num1)\*int(num2))

6. 6.group anagram

class Solution:

    def groupAnagrams(self, s: List[str]) -> List[List[str]]:

        d={}

        for i in s:

            l=list(i)

            l.sort()

            l="".join(l)

            if(l not in d):

                d[l]=[i]

            else:

                d[l].append(i)

        result=[]

        for i in d.values():

            result.append(i)

        return result

8.jump game

class Solution:

    def canJump(self, n: List[int]) -> bool:

        m=n[0]

        if(len(n)==1):

            return True

        for i in range(0,len(n)):

            if(n[i]+i>m and i<=m):

                m=n[i]+i

            if(m>=len(n)-1):

                return True

        return False

9.unique path:

class Solution:

    def uniquePaths(self, m: int, n: int) -> int:

        l=[]

        for i in range(0,m+1):

            l1=[]

            for j in range(0,n+1):

                l1.append(0)

            l.append(l1)

        for i in range(1,m+1):

            for j in range(1,n+1):

                if(i==1 and j==1):

                    l[i][j]=1

                else:

                    l[i][j]=l[i][j-1]+l[i-1][j]

        return l[-1][-1]

10 Text Justification

class Solution:

    def fullJustify(self, words: List[str], maxWidth: int) -> List[str]:

        n=len(words)

        ans=[]

        i=0

        while i<n:

            temp=[]

            seen=0

            cur=""

            while i<n and seen+len(words[i])+len(temp)<=maxWidth:

                temp.append(words[i])

                seen+=len(words[i])

                i+=1

            m=len(temp)

            if (m-1)!=0 and i!=n:

                q,r=divmod(maxWidth-seen,(m-1))

                sp=[q+(1 if j<r else 0) for j in range(m-1)]

                for j in range(m-1):

                    cur+=temp[j]+' '\*sp[j]

                cur+=temp[-1]

            else:

                for j in range(m-1):

                    cur+=temp[j]+' '

                cur+=temp[-1]

                cur+=' '\*(maxWidth-len(cur))

            ans.append(cur)

        return ans