Problema 1

**1.** O *matrice dublu sortată* este o matrice în care liniile și coloanele sunt sortate strict crescător. De exemplu, o matrice 𝑀 dublu sortată cu 𝑚=5 linii și 𝑛=4 coloane este următoarea:

1. a) Scrieți o funcție care să genereze o matrice dublu sortată de dimensiune 𝑚×𝑛 cu elemente aleatorii.
2. b) Scrieți 3 funcții, având complexitățile 𝒪(𝑚𝑛),𝒪(𝑚log2𝑛) și 𝒪(𝑚+𝑛), care să verifice dacă un număr 𝑥 se găsește sau nu într-o matrice dublu sortată. Funcția va furniza o poziție pe care apare valoarea 𝑥 în matrice, sub forma unui tuplu (linie, coloană), sau valoarea None dacă 𝑥 nu se găsește în matrice.

def matrice\_rand(n,m):  
 M = [[random.randint(0,100)] for x in range(m)]  
 M.sort()  
 for i in range(1, n):  
 M[0].extend([M[0][i - 1] + random.randint(1,10)])  
 for i in range(1, m):  
 for j in range(1, n):  
 M[i].extend([max(M[i][j - 1], M[i - 1][j]) + random.randint(1,10)])  
 return M  
  
def O\_mn(x,M,m,n):  
 for i in range(m):  
 for j in range (n):  
 if x== M[i][j]:  
 return (i,j)  
 return None

def O\_mlogn(x,M,m,n):  
 for i in range(m):  
 st=0  
 dr=n-1  
 while(st<dr):  
 m=(st+dr)//2  
 if(M[i][m]==x):  
 return i,m  
 elif(x>M[i][m]):  
 st=m+1  
 else:  
 dr=m-1  
 return None

def O\_mplusn(x,M,m,n):  
 i = 0  
 j = n-1  
 while(i>=0 and i <m and j<n and j>0):  
 if(x==M[i][j]):  
 return i,j  
 elif(x<M[i][j]):  
 j-=1  
 else:  
 i+=1  
 return None  
  
m= int(input())  
n= int(input())  
M=[[x+1] for x in range(m)]  
for i in range(1,n):  
 M[0].extend([M[0][i-1]+1])  
for i in range(1,m):  
 for j in range(1,n):  
 M[i].extend([max(M[i][j-1],M[i-1][j])+1])  
  
M2=matrice\_rand(n,m)  
for list in M2:  
 print(list)

x=M2[2][2]  
print(x)  
print(O\_mlogn(x,M2,m,n))

Problema 2

**2.** Modificați rezolvarea problemei 5 astfel încât planificarea optimă să nu mai conțină zilele libere (e.g., ziua 4 din planificarea prezentată în exemplul dat).

f = open("proiecte.in","r")  
l=[]  
for linie in f:  
 p = linie.split()  
 tup=(p[0],int(p[1]),int(p[2]))  
 l.append(tup)  
f.close()  
  
def cheieProfit(proiect):  
 return proiect[2]  
l.sort(key=cheieProfit, reverse=True)  
zi\_max = max([proiect[1] for proiect in l])  
dict\_plan = {k: None for k in range(1,zi\_max+1)}  
profit = 0  
for proiect in l:  
 for zi in range(proiect[1],0,-1):  
 if dict\_plan[zi] is None:  
 dict\_plan[zi]=proiect  
 profit += proiect[2]  
 break  
  
for zi in dict\_plan:  
 if dict\_plan[zi] is None:  
 for i in range(zi,zi\_max+1):  
 if dict\_plan[i] is not None:  
 break  
 if i<=zi\_max:  
 aux = dict\_plan[i]  
 dict\_plan[i] = dict\_plan[zi]  
 dict\_plan[zi] = aux  
  
f= open("proiecte.out","w")  
for zi in dict\_plan:  
 if dict\_plan[zi]!= None:  
 s="Ziua " + str(zi) + ": " + dict\_plan[zi][0] + " " + str(dict\_plan[zi][2]) + "\n"  
 f.write(s)  
f.write("\n Profit: "+str(profit))  
f.close()

Problema 3

**3.** În fiecare zi, 𝑛 șoferi transmit către Compania Națională de Administrare a Infrastructurii Rutiere informații referitoare la starea unei anumite autostrăzi, respectiv intervale închise de forma [𝑥,𝑦] având semnificația "*asfaltul de pe autostradă este avariat între kilometrii* 𝑥 *și* 𝑦". Considerând faptul că toate informațiile transmise de către șoferi sunt corecte și cunoscând lungimea autostrăzii respective, scrieți un program care, la sfârșitul zilei respective, să furnizeze angajaților Companiei Naționale de Administrare a Infrastructurii Rutiere următoarele informații:

1. a) porțiunile de autostradă care sunt avariate, sub forma unei reuniuni de intervale;
2. b) porțiunile de autostradă care nu sunt avariate, sub forma unei reuniuni de intervale deschise;
3. c) gradul de uzură al autostrăzii, respectiv raportul dintre numărul total de kilometri avariați de autostradă și lungimea autostrăzii.

#citirea datelor  
fin = open("autostrada.in","r")  
km\_max= int(fin.readline())  
intervale=[]  
for linie in fin:  
 l= linie.split()  
 intervale.append((int(l[0]),int(l[1])))  
  
fin.close()  
  
#sortare  
def cheie(l):  
 return l[0]  
intervale.sort(key=cheie)  
  
  
#intersectie intervale avariate  
st=0  
dr=0  
avariate=[]  
km\_av=0  
for i in range(1,len(intervale)):  
 if(intervale[i][0]<=intervale[i-1][1]):  
 if(intervale[i][1]>intervale[i-1][1]):  
 dr=i  
 else:  
 avariate.append([intervale[st][0],intervale[dr][1]])  
 km\_av+=intervale[dr][1]-intervale[st][0]  
 st=dr=i  
avariate.append([intervale[st][0],intervale[dr][1]])  
km\_av+=intervale[dr][1]-intervale[st][0]

#intervale neavariate  
liber=[]  
st=0  
for i in avariate:  
 liber.append((st,i[0]))  
 st=i[1]  
liber.append((i[1],km\_max))  
  
#afisare  
fout = open("autostrada.out","w")  
for i in avariate:  
 fout.write(str(i)+"\n")  
fout.write("\n")  
for i in liber:  
 fout.write(str(i)+"\n")  
fout.write("\n"+str(km\_av\*100//km\_max)+"%")  
fout.close()

Problema 4

Se consideră o mulțime 𝐴 formată din 𝑛+1 numere reale și un număr real 𝑥0. Considerând faptul că mulțimea 𝐴 conține cel puțin un număr real nenul, scrieți un program care să determine un polinom 𝑃(𝑥)=𝑎𝑛𝑥𝑛+𝑎𝑛−1𝑥𝑛−1+⋯+𝑎1𝑥+𝑎0 de grad 𝑛 având toate elementele mulțimii 𝐴 drept coeficienții pentru care valoarea 𝑃(𝑥0) este maximă. De exemplu, pentru 𝐴 = {2,−1,7,3} și 𝑥0=2, polinomul cerut este 𝑃(𝑥)=7𝑥3+3𝑥2+2𝑥−1.

#Tema 4  
x= float(input("x="))  
n=int(input("n="))  
A= [float(x) for x in input().split()]  
  
L=[]  
if(x>=1):  
 L=A.sort(reverse=True)  
elif(x<=-1):  
 A.sort()  
 if(n%2!=0):  
 st = 0  
 dr = n  
 for i in range(n + 1):  
 if (i % 2 == 0):  
 L.extend([A[st]])  
 st += 1  
 else:  
 L.extend([A[dr]])  
 dr -= 1  
 if(n%2==0):  
 st = 0  
 dr = n  
 for i in range(n + 1):  
 if (i % 2 != 0):  
 L.extend([A[st]])  
 st += 1  
 else:  
 L.extend([A[dr]])  
 dr -= 1  
elif(x>0):  
 L=A.sort()  
else:  
 A.sort()  
 L=[0]\*(n+1)  
 if(n%2==0):  
 p = n//2-1  
 for i in range(1, n+1, 2):  
 L[i] = A[p]  
 p -= 1  
 p = n//2  
 for i in range(0, n+1, 2):  
 L[i] = A[p]  
 p += 1  
 else:  
 p=n//2  
 for i in range(0,n,2):  
 L[i]=A[p]  
 p -= 1  
 p=n//2+1  
 for i in range(1,n+1,2):  
 L[i]=A[p]  
 p += 1  
  
print(A)  
print(L)  
print("P= ",end="")  
for i in range (n,0,-1):  
 print(L[n-i],"\* x ^",i," + ",end=" ")  
print(L[len(L)-1])