Halil İbrahim KÖSE

161044105

PART 1:

```
#dynamic programming algorithm to solve problem
a=[[1,10],[3,2]]
def takefirst(elem):#to use for sort function
    return elem[1]#while sorting elements in list we should take second element from each list
def work(L):
    L.sort(key=takefirst)#sorting list for belong to their priorities
    L.reverse()#after sorting we should reverse to find dicreasing list
   #priority list

    temp=0
    total=0
    for i in range(len(L)):#while in list
        total=((L[i][0]+temp)*L[i][1])+total
        temp+=L[i][0]

    return total

work(a)
```

Sort function time comlextiy is O(nlogn)

And for loop is O (n)

Reverse function that i used is O(n)

Time Complexty is : O (nlogn)+o(n)+o(n)=O(nlogn)

PART 2:

A:

```
FOR i= 1 TO N
    IF Ni < Si THEN
        OUTPUT "NY IN MONTH i"
    ELSE
        OUTPUT "SF IN MONTH i"
END
```

This algorithm shown below does not correctly solve this problem by giving an instance which it does not return the correct answer,because there is no movment cost part in the algorithm and no total cost.

B:

```
Ny=[1,3,50,60]
Sf=[40,30,4,5]

def findOptimal(n,m,A,B):
    moving=0
    plan=[]
    cost=0
    for i in range(n):
        if  A[i]<B[i]:
            plan.append(A[i])
            cost=cost+A[i]
            if moving==0:
                #for first city there is no movement
                moving=1
                #current city number
            if moving==2:
                #if move from other city
                moving=1
                #change the current city and
                cost=cost+m
                #sum with movement cost M


        if A[i]>B[i]:
            plan.append(B[i])
            cost=cost+B[i]
            if moving==0:
                #for first city there is no movement
                moving=2
                #current city number
            if moving==1:
                #if move from other city
                moving=2
                #change the current city and
                cost=cost+m
                #sum with movement cost M

    return ("cost of optimal plan",plan,"is",cost)


findOptimal(4,10,Ny,Sf)
```

Time complexty is O(n).