Homework 1.
Due: Monday, September, 6 2021 before 8am EDT.

Problem 1 (Jumping frog (II))

(a) Define the entries of your table in words. E.g., $T(i)$ or $T(i,j)$ is ....

Let T[i,j]= is number of ways Rene can jump from grid at ith row and jth column

(b) State recurrence for entries of your table in terms of smaller subproblems.

We will start with base case T[1,1]=1 , where Rene is located at this stone on Grid . For any other stones on grid, Rene can go to either (i+1,j+2) or (i+2,j+1) .On grid , if Rene is at either first or second row he can go to nth row. Similarly if Rene is at either first or second column of grid he can reached nth column by humping j+2 or j+1 . So we have to calculate all four grid locations from where Rene can reached at (n,n) location at Grid.

T[1][1]=1

T[i][j]=T[i-1][j-2]+T[i-2][j-1]) (when 2<= i <= n and 2 <= j <= n)

(c) Write pseudocode for your algorithm to solve this problem.

```
T[1][1]=1
for i=2 to n do
   for j=2 to n do
      T[i][j]=T[i-1][j-2]+T[i-2][j-1]
return T(n,n)
```

(d) State and analyze the running time of your algorithm.

for i=2 to n do: O(n)

for j=2 to n do: O(n)

O(n)*O(n)=O(n2)

Problem 2 (Electoral colleges)

(a) Define the entries of your table in words. E.g., $T(i)$ or $T(i,j)$ is ....

Let n be total no. of states with population pi and number of electoral votes vi.
Z is the total number of electoral votes needed to win.
for 1 <= i <=P
for 1 <=j <= C
Let T(i,j) = max value possible votes of states 1,.....,j and total number of population <=i

(b) State recurrence for entries of table in terms of smaller subproblems.

T(i,0) =0 for 0 <= i <=P
T(0 ,j) =0 for 0<= j <=C

For 1<=i<=P
For 1<=j<=C
T(i,j) = max{T (i, j-1) , T(i-v[j] ,j-1) + p[j] } if i>=v[j]
=T(i,j-1) if i<v[j]

Where C=sum(V[i])-Z
C is the sum of total votes minus total number of votes needed to win. Here we have changed the maximum capacity(votes), instead of Z to total number of votes minus Z.

(c) Write pseudocode for your algorithm to solve this problem.

```
VS=sum(V)
PS=sum(P)
C= VS -Z

for i=0 to P do
    T(i,0)=0

for j=0 to C do
    T(0,j)=0

for i=1 to P do
    For j=1 to C do
        If i>=v[j] do
                T(i,j)=max{T(i,j-1)  , T( i-v[j] ,j )     +p[j]} }
        else:
                T(i,j)=T(i,j-1)
Return PS-T(P,n)
```

(d) State and analyze the running time of your algorithm.

For i=0 to P: O(P)
For j=0 to C: O(Z)
For i=1 to P: O(P)
    For j=1 to C:O(Z)

O(P)+O(Z)+O(P)*O(Z)=O(PZ)
Total run time if O(PZ)