**//a)Implement 0/1 Knapsack problem using dynamic programming.**

#include <bits/stdc++.h>

using namespace std;

int max (int x, int y) { return (x > y)? x : y; }

int knapsack(int weight[],int profit[],int n,int W)

{ int k[n+1][W+1];

for(int i=0;i<=n;i++)

{

for(int w=0;w<=W;w++)

{

if(i==0||w==0)

k[i][w]=0;

else

{

k[i][w]=max(k[i-1][w],profit[i-1]+k[i-1][w-weight[i-1]]);

}

}

}

return k[n][W];

}

int main()

{ int n;

cout<<"Enter the number of items:"<<"\n";

cin>>n;

int weight[n],profit[n],W;

cout<<"Enter the weights of the items:"<<"\n";

for(int i=0;i<n;i++)

cin>>weight[i];

cout<<"Enter the values of the items:"<<"\n";

for(int i=0;i<n;i++)

cin>>profit[i];

cout<<"Enter the capacity of the bag:"<<"\n";

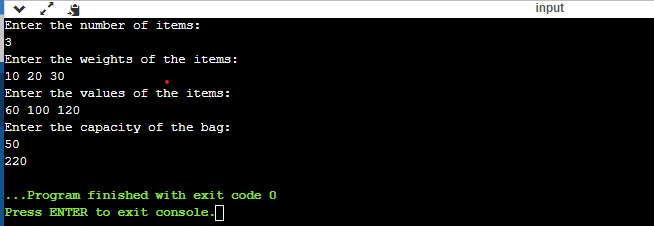
cin>>W;

int res=knapsack(weight,profit,n,W);

cout<<res;

return 0;

}



//**b) Given a sequence, find the length of the longest palindromic subsequence in it.**

#include <bits/stdc++.h>

using namespace std;

int max (int x, int y) { return (x > y)? x : y; }

int lps(char \*str)

{

int n = strlen(str);

int i, j, cl;

int L[n][n];

for (i = 0; i < n; i++)

L[i][i] = 1;

for (cl=2; cl<=n; cl++)

{

for (i=0; i<n-cl+1; i++)

{

j = i+cl-1;

if (str[i] == str[j] && cl == 2)

L[i][j] = 2;

else if (str[i] == str[j])

L[i][j] = L[i+1][j-1] + 2;

else

L[i][j] = max(L[i][j-1], L[i+1][j]);

}

}

return L[0][n-1];

}

int main()

{ char seq[20];

cout<<"enter the sequence"<<"\n";

cin>>seq;

int n = strlen(seq);

cout<<"The length of the LPS is"<<lps(seq);

return 0;

}

