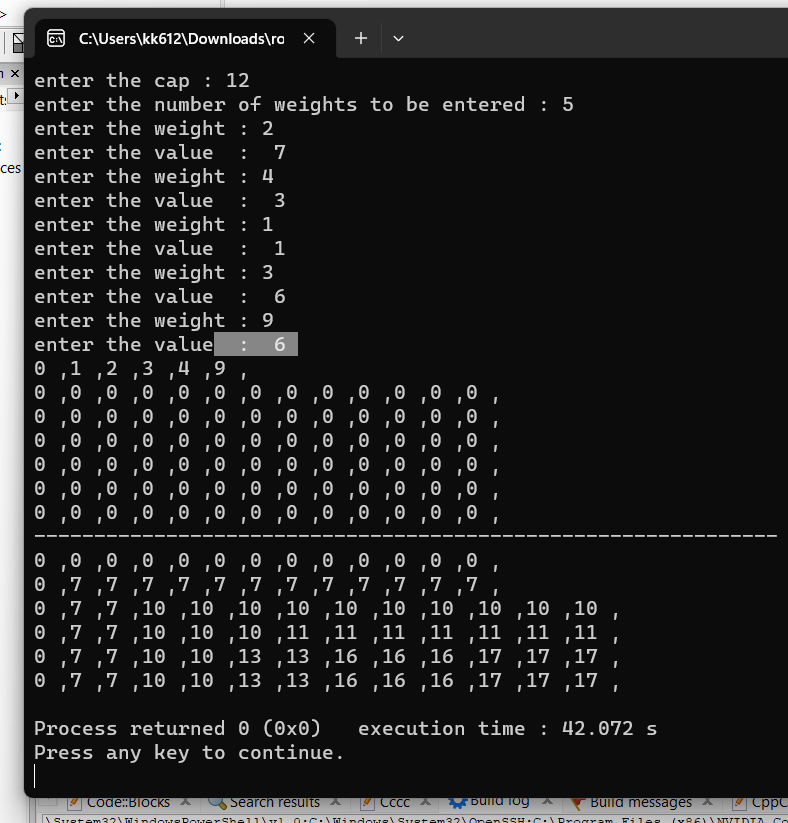
**11.) 0-1 knapsack**



#include <bits/stdc++.h>

using namespace std;

void print(int \*\*arr,int n,int cap){

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

for(int j=0;j<=cap;j++){

cout<<arr[i][j]<<" ,";

}

cout<<endl;

}

}

//this function is recursion + memoization

//here a 2-d array is used to save the already calculated optimal solution

//removes repeating recursion

int knapsack\_memoization(int \*w ,int n,int cap,int \*\*m){ // 11

if(n==0 || cap==0){

return 0;

}

if(m[n][cap] != -1){ //already found the optimum solution for current sub-problem

return m[n][cap];

}else{

if(cap < w[n] ){

return knapsack\_memoization(w,n-1,cap,m);

}

// {5 + (w,3,6) , (w,3,11) //

m[n][cap] = max( w[n] + knapsack\_memoization(w,n-1,cap-w[n],m), knapsack\_memoization(w,n-1,cap,m) );

return m[n][cap];

}

}

//this function is complete iteration based

//using a table

//calculated all possible sub - problems here , value = weight

void knapsack(int \*\*arr , int \*w ,int n,int cap,int \*v){

for(int i=1;i<=n;i++){

for(int j=1;j<=cap;j++){

if(j>= w[i]){ //now you can fill the bag , since bag is bigger than object

arr[i][j] = max( v[i]+ arr[i-1][j-w[i]] , arr[i-1][j] ) ;

}else{

arr[i][j] = arr[i-1][j]; //else take the value of previous sub problem when current item weight is larger than bag

}

}

}

print(arr,n,cap);

}

int main()

{

cout<<"enter the cap : ";

int cap=0;

cin>>cap;

cout<<"enter the number of weights to be entered : " ;

int n=0;

cin>>n;

int \*w = new int[n+1];

int \*v = new int[n+1];

w[0]=0;

v[0]=0;

int weight=0,value=0;

for(int i=1;i<=n;i++){

cout<<"enter the weight : ";

cin>>weight;

cout<<"enter the value : ";

cin>>value;

w[i] = weight;

v[i] = value;

}

sort(w, w + n+1);

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

cout<<w[i]<<" ,";

}

cout<<endl;

int \*\*arr = new int\*[n+1];

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

arr[i] = new int[cap+1];

}

for(int i=0;i<=n;i++){ //initialization of table

for(int j=0;j<=cap;j++){

arr[i][j]=0;

}

}

print(arr,n,cap);

cout<<"--------------------------------------------------------------"<<endl;

knapsack(arr,w,n,cap,v);

/\* int \*\*m= new int\*[n+1];

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

m[i] = new int[cap+1];

}

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

for(int j=0;j<=cap;j++){

m[i][j] =-1;

}

}

\*/

//cout<<knapsack\_memoization(w,n,cap,m);

return 0;

}