الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يونيبرسيتي إسلام انتارابڠسا مليسيا
Garden of Knowledge and Virtue

DATE: 06-JUN-2022

ASSIGNMENT – 2 Dynamic Programming: Programming Assignment

DATA STRUCTURES AND ALGORITHMS II

Course: CSCI-3302, Sec: 02

Lecturer: Dr. NURUL LIYANA BINTI MOHAMAD ZULKUFLI

Submitted by:

Name: Ahmed Jobaer
Matric no: 1918243

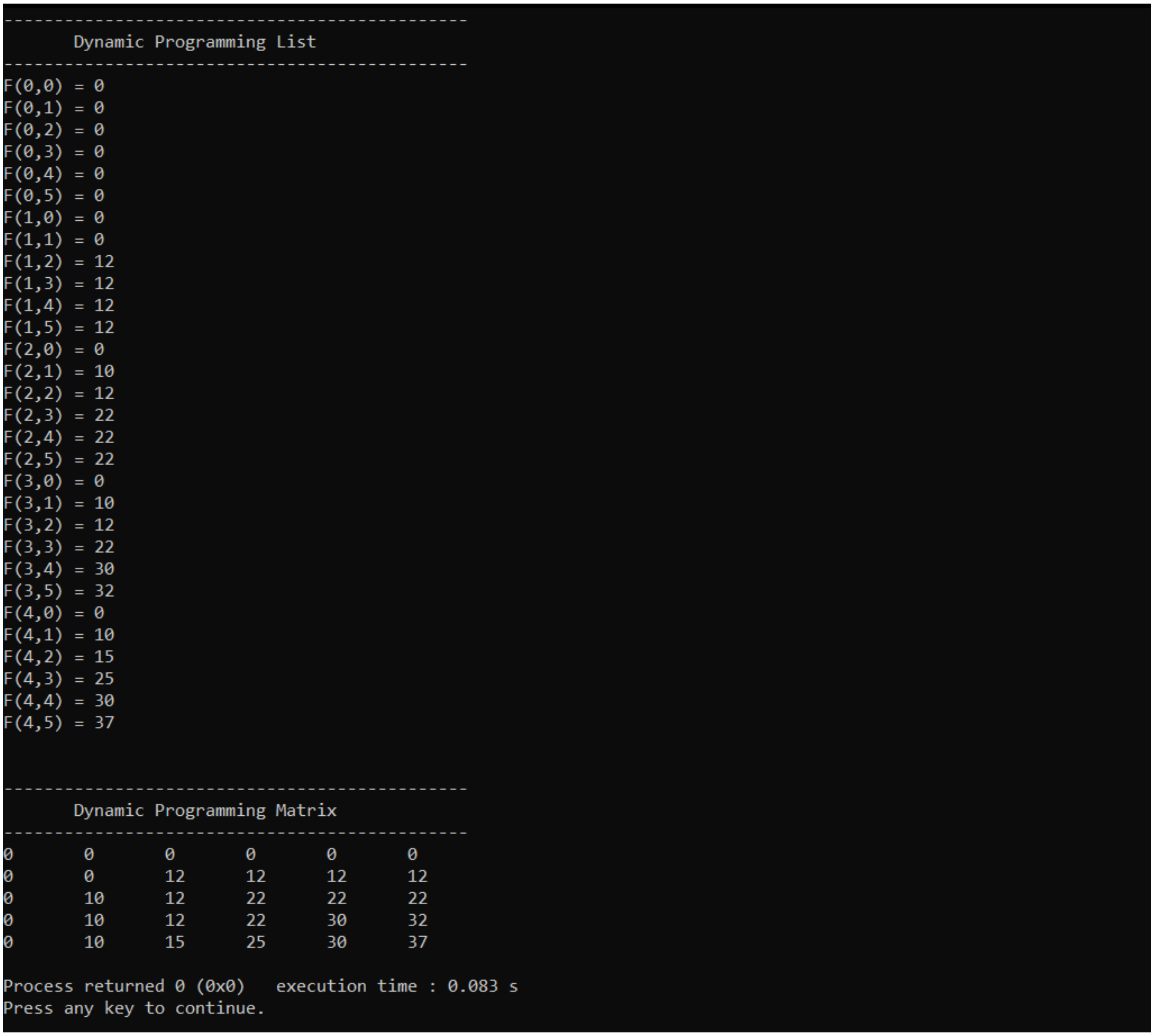
Question 1

Using the incomplete programming code given, complete the code **using dynamic programming (literately),** to **reproduce the results in the following Table 1**.

Code:

```
//================================
// Dynamic Programming - Knapsack
//================================

// Please fill up the following
// Name                         : Ahmed Jobaer
// Matric No.         : 1918243
// Section            : Sec 2
// (if 2pm-3.20pm: Sec 1; else if 3.30pm - 4.50pm: Sec 2)


//==================================================================
#include<iostream>
using namespace std;


const int W = 5; // max knapsack capacity

//========================================================================
// Dynamic Programming function for solving Knapsack Problem
void Knapsack(int i, int j, int w[], int n, int v[], int F[][W+1])
{

    for(i=0; i<=n; i++) // this for loop is depends on number of items
    {
        for(j=0; j<=W; j++) // this for loop is depends on max knapsack capacity
        {
            // if items and capacity is 0 then initialize array with 0
            if(i==0)
                F[i][j]=0;
            else if(j==0)
                F[i][j]=0;

            // here i use knapsack formula to find max value
            else if(w[i-1]<=j)
            {
                F[i][j]= max(F[i-1][j],F[i-1][j-w[i-1]]+v[i-1]);
            }
            else
                F[i][j]=F[i-1][j];
        }
    }

}

//===============
// Main Function
int main(void)
{
//===============

    // number of items
    const int n = 4;

    int i;            // item: 1 <= i <= n
    int j;            // capacity: 1 <= j <= W


    int weight[] = {2, 1, 3, 2};         // weight of each item
    int value[] = {12, 10, 20, 15};      // value of each item


    // Instruction: Initialize F
    //F: dynamic programming table/matrix, containing values of F(i,j)
    int F[n+1][W+1];


    // call the dynamic programming function
    Knapsack(i, j, weight, n, value, F);

    // show the result with list
    cout<<"----------------------------------------------"<<endl;
    cout<<"       Dynamic Programming List"<<endl;
    cout<<"----------------------------------------------"<<endl;


    for (i = 0; i <= n; i++)
    {
        for (j = 0; j <= W; j++)
        {
            cout << "F(" << i << "," << j << ") = " << F[i][j] << endl;
        }
    }

    cout << endl;


    // show the result with table
    cout<<"\n----------------------------------------------"<<endl;
    cout<<"       Dynamic Programming Matrix"<<endl;
    cout<<"----------------------------------------------"<<endl;

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

        for (j = 0; j <= W; j++)
        {
            cout<<F[i][j]<<"\t";
        }
         cout<<endl;
     }


     return 0;
 }

```

Output:



```
----------------------------------------------
       Dynamic Programming List
----------------------------------------------
F(0,0) = 0
F(0,1) = 0
F(0,2) = 0
F(0,3) = 0
F(0,4) = 0
F(0,5) = 0
F(1,0) = 0
F(1,1) = 0
F(1,2) = 12
F(1,3) = 12
F(1,4) = 12
F(1,5) = 12
F(2,0) = 0
F(2,1) = 10
F(2,2) = 12
F(2,3) = 22
F(2,4) = 22
F(2,5) = 22
F(3,0) = 0
F(3,1) = 10
F(3,2) = 12
F(3,3) = 22
F(3,4) = 30
F(3,5) = 32
F(4,0) = 0
F(4,1) = 10
F(4,2) = 15
F(4,3) = 25
F(4,4) = 30
F(4,5) = 37


----------------------------------------------
       Dynamic Programming Matrix
----------------------------------------------
0       0       0       0       0       0
0       0       12      12      12      12
0       10      12      22      22      22
0       10      12      22      30      32
0       10      15      25      30      37

Process returned 0 (0x0)   execution time : 0.083 s
Press any key to continue.
```

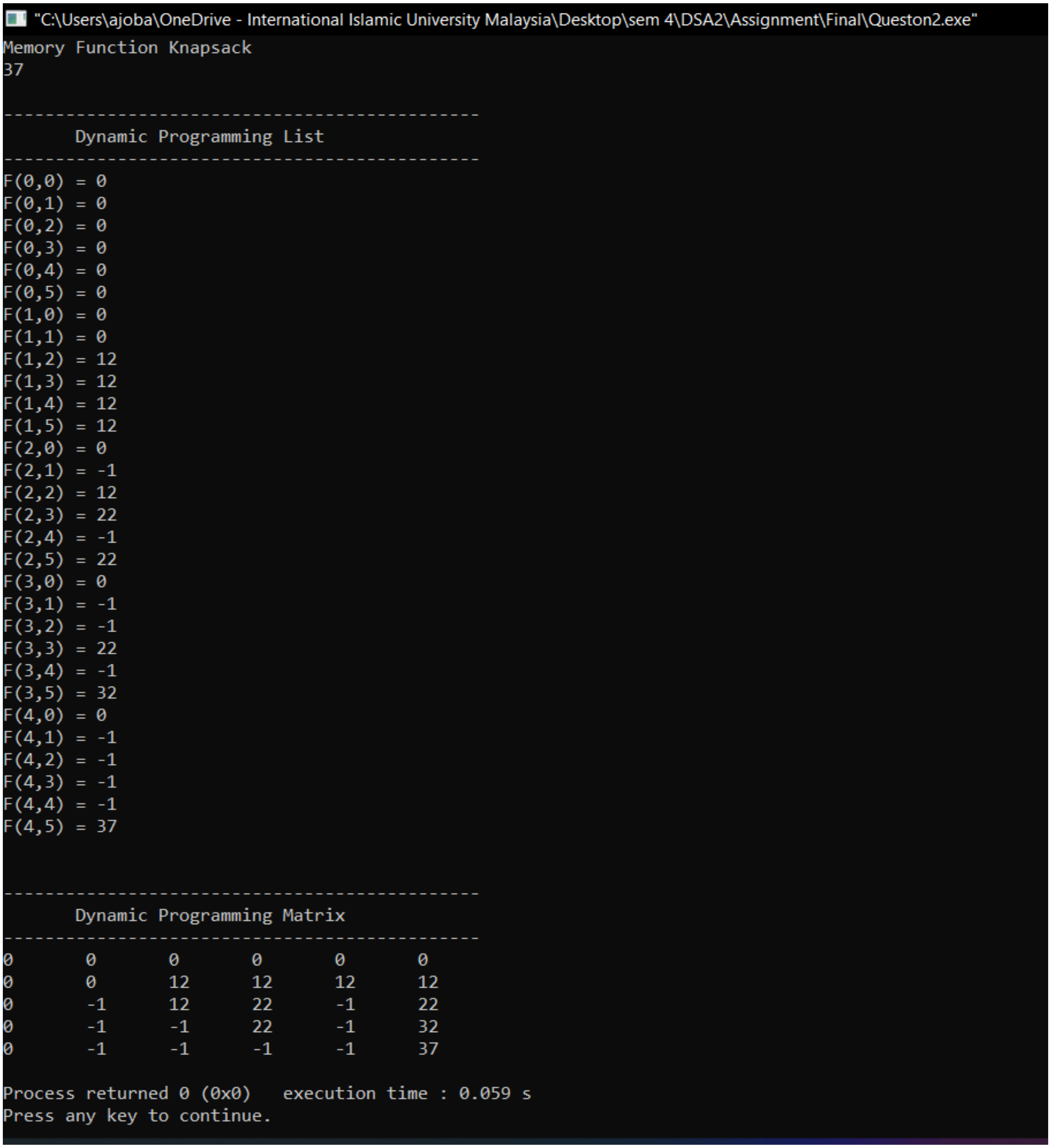
Question 2

Using the incomplete programming code given, complete the code **using dynamic programming with memory function, to reproduce the results in the following Table 1**.

Code:

```
//====================================================
// Dynamic Programming - Knapsack - Recursion + Table
//====================================================

// Please fill up the following
// Name                         : Ahmed Jobaer
// Matric No.         : 1918243
// Section            : 02
// (if 2pm-3.20pm: Sec 1; else if 3.30pm - 4.50pm: Sec 2)


#include<iostream>
using namespace std;

//Initialization
int F[5][6];
int w[5];
int v[100];

int MFKnapsack(int i,int j)
{
    // if items and capacity is 0 then initialize array with 0
    if(i==0)
        return 0;
    else if(j==0)
        return 0;

    // else then recursively calling dynamic programming function
    else if(w[i]<=j)
        F[i][j] = max(MFKnapsack(i-1,j),v[i]+MFKnapsack(i-1,j-w[i]));
    else
        F[i][j] = MFKnapsack(i-1,j);
    return F[i][j];
}


int main(void)
{

    int i; // item: 1 <= i <= n  (row)
    int j; // capacity: 1 <= j <= W (column)


    // Initialization

    // start from the goal F(n, W), then recursively calling dynamic programming function
    i = 4;
    j = 5;

    int weight[] = {2,1,3,2};   // weight of each item
    int value[] = {12,10,20,15};         // value of each item

    for(int ii=0; ii<=i; ii++)
    {
        // if number of items and capacity is 0 then initialize F with 0
        for(int jj=0; jj<=j; jj++)
        {
            if(ii==0 || jj==0)
                F[ii][jj]=0;
        // otherwise it -1
            else
                F[ii][jj]=-1;
        }
    }

    // store the value in array
    for(int ii=0; ii<i; ii++)
    {
        w[ii+1]=weight[ii];
        v[ii+1]=value[ii];
    }

    cout << "Memory Function Knapsack \n";
    cout << MFKnapsack(i,j) << endl;


    // show the result

    cout<<"\n----------------------------------------------"<<endl;
    cout<<"       Dynamic Programming List"<<endl;
    cout<<"----------------------------------------------"<<endl;


    for (int ii = 0; ii <=i; ii++)
    {
        for (int jj = 0; jj <=j; jj++)
        {
            cout << "F(" << ii << "," << jj << ") = " << F[ii][jj] << endl;
        }
    }

    cout << endl;


    cout<<"\n----------------------------------------------"<<endl;
    cout<<"       Dynamic Programming Matrix"<<endl;
    cout<<"----------------------------------------------"<<endl;

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

         for (int jj = 0; jj <= j; jj++)
         {
             cout<<F[ii][jj]<<"\t";
         }
         cout<<endl;
     }


     return 0;
 }

```

Output:



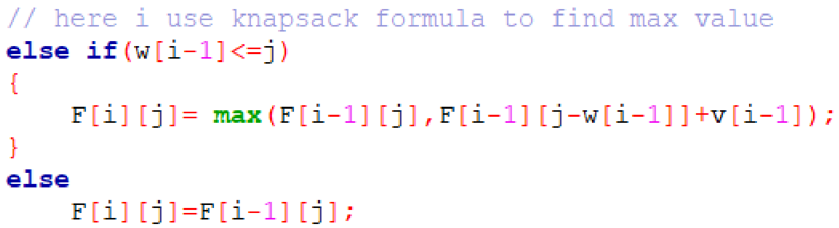
"C:\Users\ajoba\OneDrive - International Islamic University Malaysia\Desktop\sem 4\DSA2\Assignment\Final\Queston2.exe"

```
Memory Function Knapsack
37

----------------------------------------------
       Dynamic Programming List
----------------------------------------------
F(0,0) = 0
F(0,1) = 0
F(0,2) = 0
F(0,3) = 0
F(0,4) = 0
F(0,5) = 0
F(1,0) = 0
F(1,1) = 0
F(1,2) = 12
F(1,3) = 12
F(1,4) = 12
F(1,5) = 12
F(2,0) = 0
F(2,1) = -1
F(2,2) = 12
F(2,3) = 22
F(2,4) = -1
F(2,5) = 22
F(3,0) = 0
F(3,1) = -1
F(3,2) = -1
F(3,3) = 22
F(3,4) = -1
F(3,5) = 32
F(4,0) = 0
F(4,1) = -1
F(4,2) = -1
F(4,3) = -1
F(4,4) = -1
F(4,5) = 37


----------------------------------------------
       Dynamic Programming Matrix
----------------------------------------------
0       0       0       0       0       0
0       0       12      12      12      12
0       -1      12      22      -1      22
0       -1      -1      22      -1      32
0       -1      -1      -1      -1      37

Process returned 0 (0x0)   execution time : 0.059 s
Press any key to continue.
```

Question 3

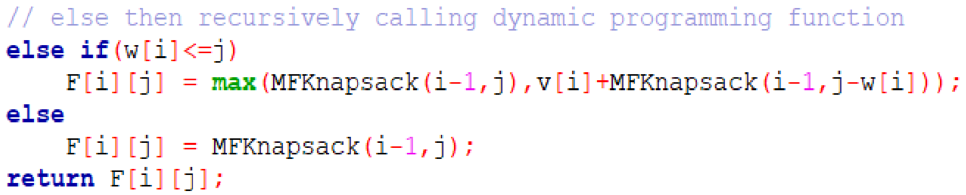
Explain in your own simple words, how your codes in Question 1 and 2 are similar but different to each other. Show where they differ/similar.

a. Fastly, both have same input, but calculation approach is different.
b. Both **Question 1 & Question 2,** if weight or capacity is 0 then I initialize the array with 0. Besides, in **Question 2,** if weight or capacity is not 0 then I initialize the array with -1.
c. Main difference is:



```
// here i use knapsack formula to find max value
else if(w[i-1]<=j)
{
    F[i][j]= max(F[i-1][j],F[i-1][j-w[i-1]]+v[i-1]);
}
else
    F[i][j]=F[i-1][j];
```

Iteratively



```
// else then recursively calling dynamic programming function
else if(w[i]<=j)
    F[i][j] = max(MFKnapsack(i-1,j),v[i]+MFKnapsack(i-1,j-w[i]));
else
    F[i][j] = MFKnapsack(i-1,j);
return F[i][j];
```

Recursively

Question 4

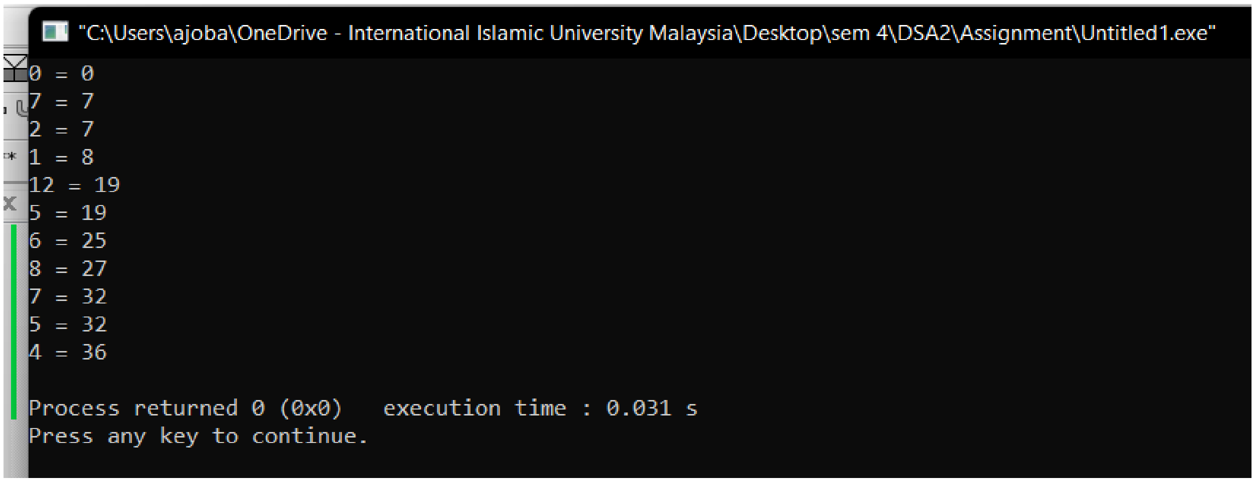
Based on your previous assignment, write your own code **to solve the following modified coin-row problem.** Use the following instance**: 7, 2, 1, 12, 5, 6, 8, 7, 5, 4.**

Code:

```
// Please fill up the following
// Name                         : Ahmed Jobaer
// Matric No.         : 1918243
// Section            : 02
// (if 2pm-3.20pm: Sec 1; else if 3.30pm - 4.50pm: Sec 2)

#include<iostream>
using namespace std;
```

```

void CoinRow(int n[], int);

//main function
int main()
{
    //Initialization
    int coins[] = { 7, 2, 1, 12, 5, 6, 8, 7, 5, 4};
    int s = sizeof(coins) / sizeof(coins[0]); // calculate the size of array

    CoinRow(coins, s); // calling function

    return 0;
}

void CoinRow(int arr[],int arrsize)
{

    int C[arrsize + 1];
    for(int i = 0; i < arrsize+1; i++)
    {
        C[i+1] = arr[i]; // copy the given elements in array C
    }

    //Initialization
    int F[arrsize + 1];
    F[0] = 0;
    F[1] = C[1];

    for(int i = 2; i<= arrsize; i++)
    {
        F[i] = max(C[i] + F[i - 2], F[i-1]); // using recurrence formula to find maximum value
    }

    //showing the result in a list
    for(int i = 0; i<arrsize+1; i++)
    {
        cout<<arr[i-1]<<" = "<<F[i]<<endl;
    }

}

```

Output:



"C:\Users\ajoba\OneDrive - International Islamic University Malaysia\Desktop\sem 4\DSA2\Assignment\Untitled1.exe"

```
0 = 0
7 = 7
2 = 7
1 = 8
12 = 19
5 = 19
6 = 25
8 = 27
7 = 32
5 = 32
4 = 36

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

Reference:

1. The slide that you are given in Italeem.
2. https://stackoverflow.com/questions/14103846/trying-to-figure-out-the-classic-knapsack-recurrence
3. https://www.geeksforgeeks.org/coin-change-dp-7/

Sign the pledge (copy/rewrite in your simple report):

“In the name of Allah, I affirm that I did not give or receive any unauthorized help on this exam, and that all work will be my own.”

____JOBAER______