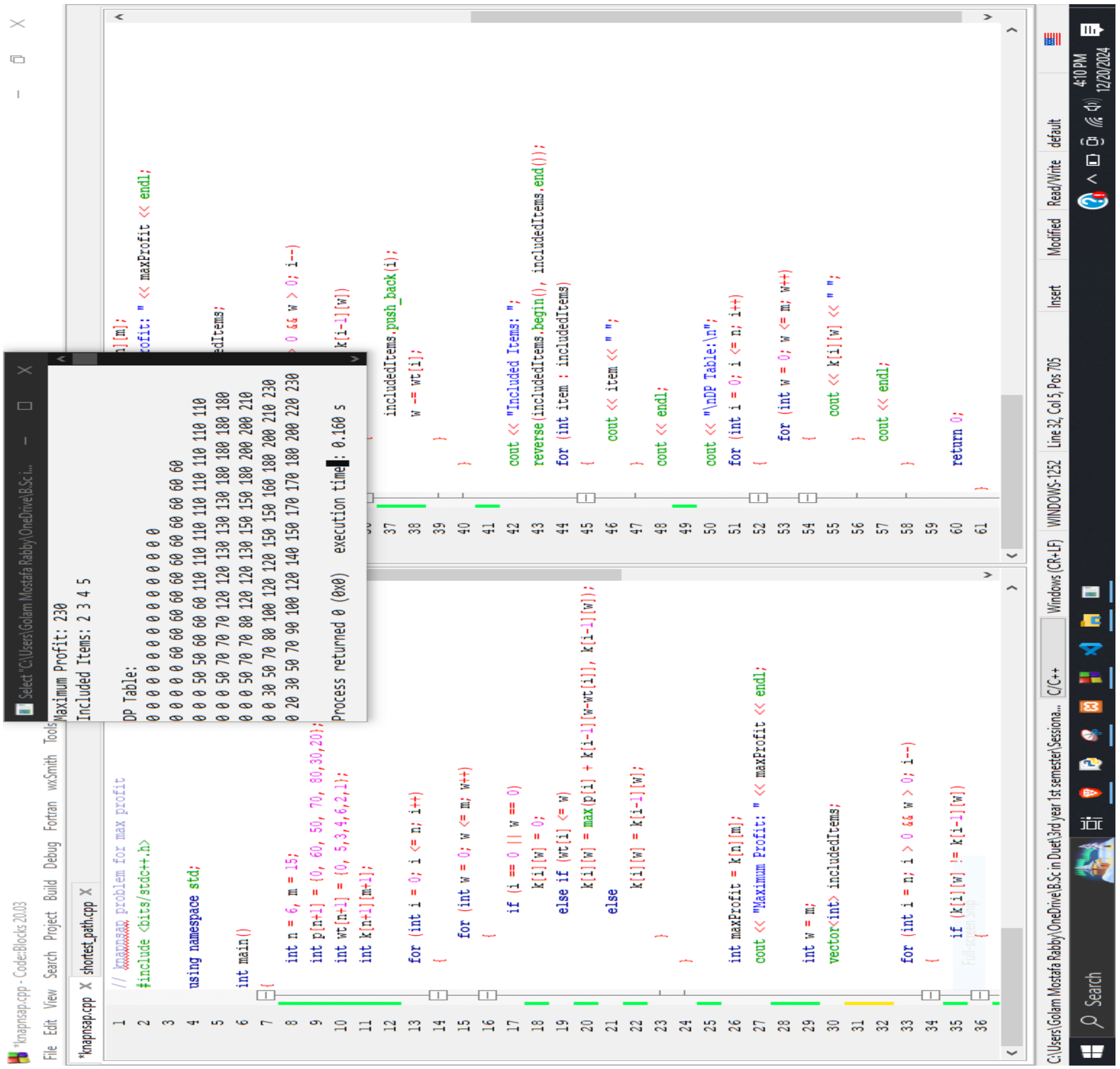
1. 0/1 Knapsack Problem: Packing a Survival Kit backpack
   You are preparing a survival kit backpack for a hiking trip. The backpack can hold a maximum weight of 15kg, and you have a list of items with their respective weights and survival values (how useful they are). Determine which items to take so that the total weight does not exceed 15 kg and the survival value is maximized. Write a C++ program to implement 0/1 Knapsack algorithm to compute the best combination of items for maximum survival value.

   You must choose from the following items:

Item	**Weight (kg)**	**Survival Value**
First Aid Kit	5	60
Water Bottle	3	50
Food Supplies	4	70
Warm Clothes	6	80
Flashlight	2	30
Map & Compass	1	20



```
// knapnsap problem for max profit
#include <bits/stdc++.h>

using namespace std;

int main()
{
    int n = 6, m = 15;
    int p[n+1] = {0, 60, 50, 70, 80,30,20};
    int wt[n+1] = {0, 5,3,4,6,2,1};
    int k[n+1][m+1];

    for (int i = 0; i <= n; i++)
    {
        for (int w = 0; w <= m; w++)
        {
            if (i == 0 || w == 0)
                k[i][w] = 0;
            else if (wt[i] <= w)
                k[i][w] = max(p[i] + k[i-1][w-wt[i]], k[i-1][w]);
            else
                k[i][w] = k[i-1][w];
        }
    }

    int maxProfit = k[n][m];
    cout << "Maximum Profit: " << maxProfit << endl;

    int w = m;
    vector<int> includedItems;


    for (int i = n; i > 0 && w > 0; i--)
    {
        if (k[i][w] != k[i-1][w])
        {
```

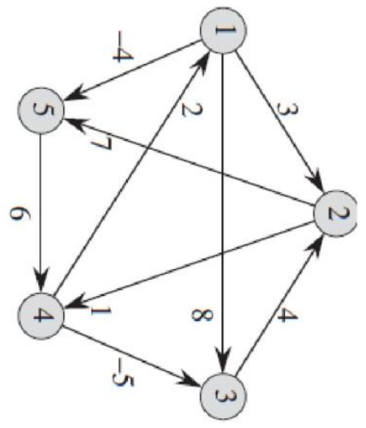
```
n][m];
rofit: " << maxProfit << endl;

edItems;

> 0 && w > 0; i--)

k[i-1][w])
```

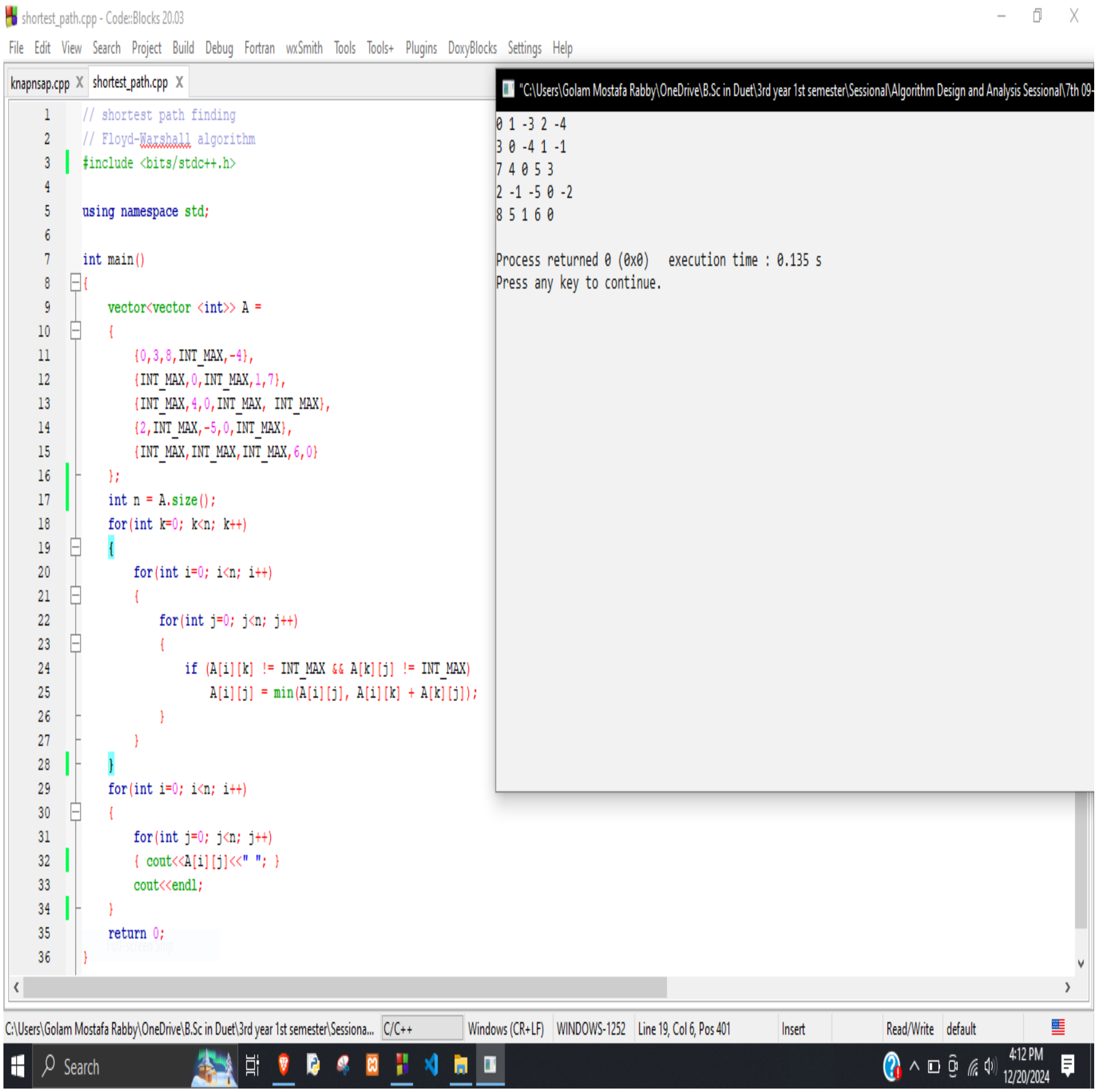
```
            includedItems.push_back(i);
            w -= wt[i];
        }
    }

    cout << "Included Items: ";
    reverse(includedItems.begin(), includedItems.end());
    for (int item : includedItems)
    {
        cout << item << " ";
    }
    cout << endl;

    cout << "\nDP Table:\n";
    for (int i = 0; i <= n; i++)
    {
        for (int w = 0; w <= m; w++)
        {
            cout << k[i][w] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

```
Maximum Profit: 230
Included Items: 2 3 4 5

DP Table:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 60 60 60 60 60 60 60 60 60 60 60
0 0 0 50 50 60 60 60 110 110 110 110 110 110 110 110
0 0 0 50 70 70 70 120 120 130 130 130 180 180 180 180
0 0 0 50 70 70 80 120 120 130 150 150 180 200 200 210
0 0 30 50 70 80 100 120 120 150 150 160 180 200 210 230
0 20 30 50 70 90 100 120 140 150 170 170 180 200 220 230

Process returned 0 (0x0)   execution time : 0.160 s
```

*knapnsap.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools
*knapnsap.cpp
shortest_path.cpp
Select "C:\Users\Golam Mostafa Rabby\OneDrive\B.Sc i...
C:\Users\Golam Mostafa Rabby\OneDrive\B.Sc in Duet\3rd year 1st semester\Sessiona...
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 32, Col 5, Pos 705
Insert
Modified
Read/Write
default
Search
4:10 PM
12/20/2024

2. Write a C++ program to implement Floyd-Warshall algorithm to find the shortest paths between all vertices in the following directed weighted graph.




shortest_path.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
knapnsap.cpp
shortest_path.cpp

```cpp
// shortest path finding
// Floyd-Warshall algorithm
#include <bits/stdc++.h>

using namespace std;

int main()
{
    vector<vector <int>> A =
    {
        {0,3,8,INT_MAX,-4},
        {INT_MAX,0,INT_MAX,1,7},
        {INT_MAX,4,0,INT_MAX, INT_MAX},
        {2,INT_MAX,-5,0,INT_MAX},
        {INT_MAX,INT_MAX,INT_MAX,6,0}
    };
    int n = A.size();
    for(int k=0; k<n; k++)
    {
        for(int i=0; i<n; i++)
        {
            for(int j=0; j<n; j++)
            {
                if (A[i][k] != INT_MAX && A[k][j] != INT_MAX)
                    A[i][j] = min(A[i][j], A[i][k] + A[k][j]);
            }
        }
    }
    for(int i=0; i<n; i++)
    {
        for(int j=0; j<n; j++)
        { cout<<A[i][j]<<" "; }
        cout<<endl;
    }
    return 0;
}
```

"C:\Users\Golam Mostafa Rabby\OneDrive\B.Sc in Duet\3rd year 1st semester\Sessional\Algorithm Design and Analysis Sessional\7th 09-

```
0 1 -3 2 -4
3 0 -4 1 -1
7 4 0 5 3
2 -1 -5 0 -2
8 5 1 6 0

Process returned 0 (0x0)   execution time : 0.135 s
Press any key to continue.
```

C:\Users\Golam Mostafa Rabby\OneDrive\B.Sc in Duet\3rd year 1st semester\Sessiona... C/C++ Windows (CR+LF) WINDOWS-1252 Line 19, Col 6, Pos 401 Insert Read/Write default