Assignment No. 11

Name – Manish Namdev Barage

PRN – 22520007

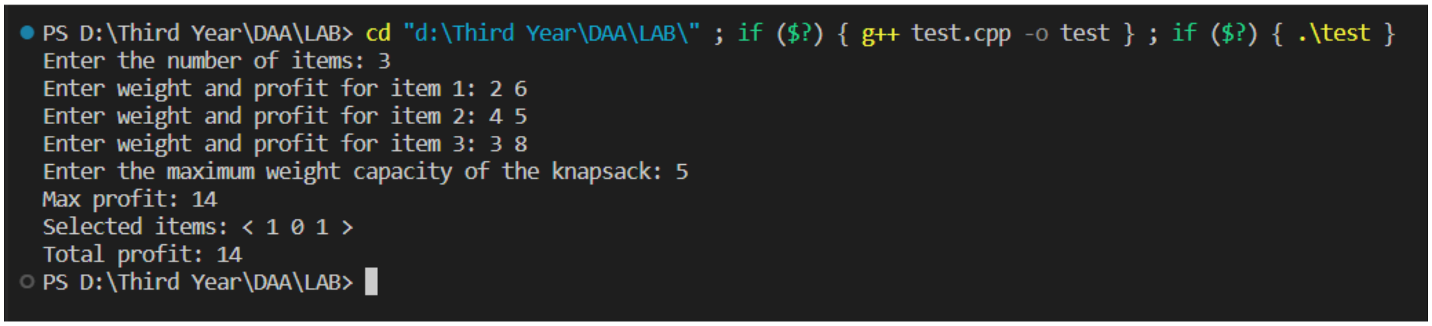
Batch – T7

Problem Statement – To Implement 0/1 Knapsack problem using dynamic programming

Code –

```
#include <iostream>
#include <vector>

using namespace std;

// Structure to represent an item
struct Item
{
    int weight;
    int profit;
};

// Function to print the selected items and their total profit
void printSelectedItems(const vector<vector<int>> &dp, const
vector<Item> &items, int n, int W)
{
    int i = n, w = W;
    vector<bool> selected(n, false);

    while (i > 0 && w > 0)
    {
        if (dp[i][w] != dp[i - 1][w])
        {
            selected[i - 1] = true;
            w -= items[i - 1].weight;
        }
        i--;
    }

    cout << "Selected items: < ";
    for (int i = 0; i < n; i++)
    {
        cout << (selected[i] ? "1" : "0");
        if (i < n - 1)
        {
            cout << " ";
        }
    }
    cout << " >" << endl;
    cout << "Total profit: " << dp[n][W] << endl;
}
```

```
// Function to solve the 0/1 knapsack problem
void knapsack01(const vector<Item> &items, int n, int W)
{
    vector<vector<int>> dp(n + 1, vector<int>(W + 1, 0));

    for (int i = 1; i <= n; i++)
    {
        for (int w = 1; w <= W; w++)
        {
            if (items[i - 1].weight <= w)
            {
                int withCurrentItem = items[i - 1].profit + dp[i -
1][w - items[i - 1].weight];
                int withoutCurrentItem = dp[i - 1][w];
                dp[i][w] = max(withCurrentItem, withoutCurrentItem);
            }
            else
            {
                dp[i][w] = dp[i - 1][w];
            }
        }
    }

    cout << "Max profit: " << dp[n][W] << endl;
    printSelectedItems(dp, items, n, W);
}

int main()
{
    int n; // Number of items
    cout << "Enter the number of items: ";
    cin >> n;

    vector<Item> items(n);

    for (int i = 0; i < n; i++)
    {
        cout << "Enter weight and profit for item " << i + 1 << ": ";
        cin >> items[i].weight >> items[i].profit;
    }

    int W; // Maximum weight capacity
    cout << "Enter the maximum weight capacity of the knapsack: ";
    cin >> W;

    knapsack01(items, n, W);

    return 0;
}
```



```
PS D:\Third Year\DAA\LAB> cd "d:\Third Year\DAA\LAB\" ; if ($?) { g++ test.cpp -o test } ; if ($?) { .\test }
Enter the number of items: 3
Enter weight and profit for item 1: 2 6
Enter weight and profit for item 2: 4 5
Enter weight and profit for item 3: 3 8
Enter the maximum weight capacity of the knapsack: 5
Max profit: 14
Selected items: < 1 0 1 >
Total profit: 14
PS D:\Third Year\DAA\LAB>
```