Assignment 4 –

#include <iostream>

#include <vector>

using namespace std;

// Function to solve 0/1 Knapsack problem

int knapsack(vector<int>& weights, vector<int>& values, int capacity) {

int n = weights.size();

// Create a 2D array to store the solutions to subproblems

vector<vector<int>> dp(n + 1, vector<int>(capacity + 1, 0));

// Fill the dp array

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

for (int w = 1; w <= capacity; ++w) {

// If the current item can be included

if (weights[i - 1] <= w) {

dp[i][w] = max(dp[i - 1][w], values[i - 1] + dp[i - 1][w - weights[i - 1]]);

} else {

// If the current item cannot be included

dp[i][w] = dp[i - 1][w];

}

}

}

// The final result is stored in dp[n][capacity]

return dp[n][capacity];

}

int main() {

// Example usage

vector<int> weights = {2, 4, 6, 9};

vector<int> values = {10, 10, 12, 18};

int capacity = 15;

int result = knapsack(weights, values, capacity);

cout << "Maximum value in knapsack: " << result << endl;

return 0;

}