**Practical No : 5**

**Name :** Samrudhi Pawar

**Roll No :** 44

**Title :**Develop a program to implement 0/1 Knapsack problem using Dynamic Programming.

**Program :**

#include <iostream>

using namespace std;

// Function to find the maximum value that can be put in a knapsack of capacity W

int knapsack(int W, int wt[], int val[], int n) {

int dp[n+1][W+1];

// Build the table dp[][] in a bottom-up manner

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

for (int w = 0; w <= W; w++) {

if (i == 0 || w == 0) {

dp[i][w] = 0; // Base case: no items or weight capacity is zero

} else if (wt[i-1] <= w) {

// Item i can be included in the knapsack

dp[i][w] = max(val[i-1] + dp[i-1][w-wt[i-1]], dp[i-1][w]);

} else {

// Item i cannot be included in the knapsack

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

}

}

}

// The last cell of dp[n][W] will have the answer (maximum value)

return dp[n][W];

}

int main() {

int n, W;

// Taking input for number of items

cout << "Enter the number of items: ";

cin >> n;

// Arrays for weights and values

int wt[n], val[n];

// Taking input for values and weights of items

cout << "Enter the values of the items:\n";

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

cin >> val[i];

}

cout << "Enter the weights of the items:\n";

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

cin >> wt[i];

}

// Taking input for the capacity of the knapsack

cout << "Enter the capacity of the knapsack: ";

cin >> W;

// Call the knapsack function and display the result

int maxVal = knapsack(W, wt, val, n);

cout << "Maximum value in the knapsack = " << maxVal << endl;

return 0;

}

**Output :**

Enter the number of items: 3

Enter the values of the items:

60

100

120

Enter the weights of the items:

10

20

30

Enter the capacity of the knapsack: 50

Maximum value in the knapsack = 220