21BRS1296 – Anika Kamath

Design and Analysis of Algorithms (Lab)

L37+L38

Experiment No.: 7

Task Number: 2

Q. Implementation of the 0/1 KnapSack Problem using Top Down Approach

Code:

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

int knapSackUtil(int W, int w[], int v[], int n, vector<vector<int>>&
dp) {
   if (n == 0 || W == 0)
       return 0;

   if (dp[n][W] != -1)
       return dp[n][W];

   if (w[n - 1] > W)
       return dp[n][W] = knapSackUtil(W, w, v, n - 1, dp);

   int include = v[n - 1] + knapSackUtil(W - w[n - 1], w, v, n - 1,
dp);
   int exclude = knapSackUtil(W, w, v, n - 1, dp);

   return dp[n][W] = max(include, exclude);
```

```
}

int knapSack(int W, int w[], int v[], int n) {
   vector<vector<int>> dp(n + 1, vector<int>(W + 1, -1));
   return knapSackUtil(W, w, v, n, dp);
}

int main() {
   cout << "Enter the capacity of the knapsack: ";
   int W;
   cin >> W;

   cout << "Enter the total number of items: ";
   int n;
   cin >> n;

   int v[n], w[n];
   cout << "Enter the weights of the items: ";
   for (int i = 0; i < n; i++) {
       cin >> w[i];
   }

   cout << "Enter the values of the items: ";
   for (int i = 0; i < n; i++) {
       cin >> v[i];
   }

   cout << "Maximum value: " << knapSack(W, w, v, n) << endl;
```

```
    return 0;
}
```

Output: