Date: 13-10-2025

Experiment No: 12

I. Aim: Implement solution for 0 / 1 knapsack problem using dynamic programming.

II. Theory:

Show stepwise procedure to find maximum weight that can be stored in sack and also list all objects that can be put inside the sack.

Profit = {3,4,5,6}
Weight = {2,3,4,5}
Sack Capacity= 8
N=4

Procedure:

_____
_____
_____
_____

III. Program:

Find maximum weight that can be stored in sack using dynamic approach:

```
#include<stdio.h>
#include<conio.h>
int max(int a, int b)
{
 if (a > b)
 return a;
 return b;
}

int knapsackRecursive(int W, int wt[], int val[], int n)
{
 if (n == 0 || W == 0)
 return 0;
 if (wt[n - 1] > W)
 return knapsackRecursive(W, wt, val, n - 1);
 else
 return max(val[n - 1] + knapsackRecursive(W - wt[n - 1], wt, val, n - 1),
 knapsackRecursive(W, wt, val, n - 1));
}
```

```
void main()
{
 int profit[] = { 2, 3, 4, 1};
 int weight[] = { 3, 4, 5, 6};
 int W = 8,n;
 clrscr();
 n = sizeof(profit) / sizeof(profit[0]);
 printf("Maximum value that can be put in knapsack: %d\n",
 knapsackRecursive(W, weight, profit, n));
 getch();
}
```

IV. Output:

V. Complexity:

Number of objects = n
Total capacity of sack = w
Time complexity = O (n*w)

VI. Conclusion: Successfully implemented algorithm to find maximum weight that can be stored in sack.