**LAB 13**

Implementation of 0-1 Knapsack Algorithm

Name : Rahul Thapar

ID : 1410110321

**CODE :**

/\*

@author : Rahul Thapar

ID : 1410110321

Implementation of 0-1 Knapsack

------------------------------

\*/

#include <stdio.h>

#include <stdlib.h>

int solution[10][10];

int keep[10][10];

int compute\_max\_val(int weight[10], int value[10],

int no\_items, int max\_weight);

void print\_optimal(int weight[10], int no\_items, int max\_weight);

int main(int argc, char \*argv[])

{

int i, no\_items, max\_weight, max\_val;

int weight[10], value[10];

printf("\t =================================\n");

printf("\t 0-1 Knapsack Implementation\n");

printf("\t =================================\n\n");

printf("\n\t Number of items : ");

scanf("%d",&no\_items);

printf("\t Maximum weight : ");

scanf("%d",&max\_weight);

printf("\t Enter %d items with their values : \n\n", no\_items);

for (i = 1; i <= no\_items; i++) {

printf("\tWeight %d :",i);

scanf("%d",&weight[i]);

printf("\t Value : ");

scanf("%d",&value[i]);

printf("\n");

}

max\_val = compute\_max\_val(weight, value, no\_items, max\_weight);

printf("\t The maximum value is: %d\n", max\_val);

print\_optimal(weight, no\_items, max\_weight);

}

int compute\_max\_val(int weight[10], int value[10],

int no\_items, int max\_weight)

{

int i, j, val1, val2;

for (i = 0; i <= max\_weight; i++) {

solution[0][i] = 0;

keep[0][i] = 0;

}

for (i = 0; i <= no\_items; i++) {

keep[i][0] = 0;

solution[i][0] = 0;

}

/\* i - no of items, j - weight \*/

for (i = 1; i <= no\_items; i++) {

for (j = 1; j <= max\_weight; j++) {

if (weight[i] > j) {

solution[i][j] = solution[i-1][j];

keep[i][j] = 0;

} else {

val1 = solution[i-1][j]; // Not choosing the i-th element

val2 = solution[i-1][j-weight[i]] + value[i]; // Choosing the i-th

// element

if (val1 >= val2) {

solution[i][j] = val1;

keep[i][j] = 0;

} else {

solution[i][j] = val2;

keep[i][j] = 1;

}

}

/\* printf("solution[%d][%d] = %d\n", i, j, solution[i][j]); \*/

}

}

return solution[no\_items][max\_weight];

}

void print\_optimal(int weight[], int no\_items, int max\_weight)

{

int i = no\_items, w = max\_weight; // i - item no, w - weight

int optimal\_soln[10] = {0};

while (i > 0) {

if (keep[i][w] == 1) {

optimal\_soln[i] = 1;

w = w - weight[i];

}

i--;

}

printf("\t The optimal solution is: ");

for (i = 0; i <= no\_items; i++) {

if (optimal\_soln[i] == 1)

printf("%d -> ", i);

}

printf("\n");

}

**SCEENSHOTS:**

