

LAB PROGRAM 7

AIM: Implement 0/1 Knapsack problem using dynamic programming.

SOURCE CODE

```
#include <stdio.h>

int max(int a, int b) {
    return (a > b) ? a : b;
}

void knapsack(int n, int weight[], int value[], int capacity) {
    int dp[n + 1][capacity + 1];

    for (int i = 0; i <= n; i++) {
        for (int w = 0; w <= capacity; w++) {
            if (i == 0 || w == 0) {
                dp[i][w] = 0;
            } else if (weight[i - 1] <= w) {
                dp[i][w] = max(value[i - 1] + dp[i - 1][w - weight[i - 1]], dp[i - 1][w]);
            } else {
                dp[i][w] = dp[i - 1][w];
            }
        }
    }

    // Print the DP table
    printf("DP Table:\n");
    for (int i = 0; i <= n; i++) {
        for (int w = 0; w <= capacity; w++) {
```

```
        printf("%d\t", dp[i][w]);
    }
    printf("\n");
}

// Backtrack to find the items included in the knapsack
printf("Items included:\n");
int i = n, w = capacity;
while (i > 0 && w > 0) {
    if (dp[i][w] != dp[i - 1][w]) {
        printf("Item %d (Value: %d, Weight: %d)\n", i, value[i - 1], weight[i - 1]);
        w -= weight[i - 1];
    }
    i--;
}
}

int main() {
    int n, capacity;

    printf("Enter the number of items: ");
    scanf("%d", &n);

    int weight[n], value[n];

    printf("Enter the weight and value of each item:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d%d", &weight[i], &value[i]);
    }

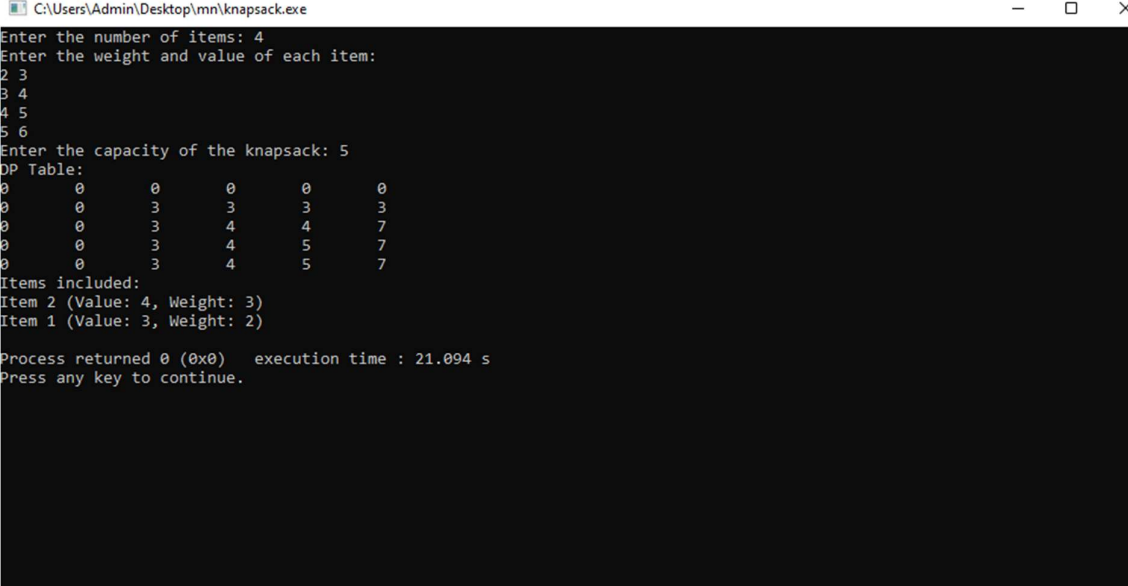
    printf("Enter the capacity of the knapsack: ");
```

```
scanf("%d", &capacity);

knapsack(n, weight, value, capacity);

return 0;
}
```

OUTPUT SCREENSHOT



```
C:\Users\Admin\Desktop\mn\knapsack.exe
Enter the number of items: 4
Enter the weight and value of each item:
2 3
3 4
4 5
5 6
Enter the capacity of the knapsack: 5
DP Table:
0 0 0 0 0
0 0 3 3 3
0 0 3 4 4
0 0 3 4 5
0 0 3 4 5
Items included:
Item 2 (Value: 4, Weight: 3)
Item 1 (Value: 3, Weight: 2)
Process returned 0 (0x0)   execution time : 21.094 s
Press any key to continue.
```