Aim: write a program in c to implement 0/1 knapsack problem using dynamic programming

Algorithm:

1. Initialize a 2D array K[n+1][W+1] where n is the number of items and W is the capacity of the knapsack.
2. Loop through the items i and the weight w of the knapsack.
3. If the weight of the current item wt[i-1] is less than or equal to w, then store the maximum value of either including the current item or not including it. K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w])
4. If the weight of the current item wt[i-1] is more than w, then the current item can't be included and the value remains the same as the previous value. K[i][w] = K[i-1][w]
5. Return K[n][W] as the maximum value that can be put in the knapsack.

Source Code:

#include <stdio.h>

#include <stdlib.h>

*int* max(*int* a, *int* b)

{

    return (a > b) ? a : b;

}

*int* knapSack(*int* W, *int* wt[], *int* val[], *int* n)

{

*int* i, w;

*int* K[n + 1][W + 1];

    for (i = 0; i <= n; i++)

    {

        for (w = 0; w <= W; w++)

        {

            if (i == 0 || w == 0)

                K[i][w] = 0;

            else if (wt[i - 1] <= w)

                K[i][w] = max(val[i - 1] +

                    K[i - 1][w - wt[i - 1]], K[i - 1][w]);

            else

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

        }

    }

    return K[n][W];

}

*int* main()

{

*int* n, W;

    printf("Enter number of items: ");

    scanf("%d", &n);

*int* val[n], wt[n];

    for (*int* i = 0; i < n; i++) {

        printf("Enter value and weight for item %d: ", i + 1);

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

    }

    printf("Enter knapsack capacity: ");

    scanf("%d", &W);

    printf("The maximum value that can be put in knapsack is %d\n", knapSack(W, wt, val, n));

    return 0;

}

Output:

Enter number of items: 3

Enter value and weight for item 1: 60 10

Enter value and weight for item 2: 100 20

Enter value and weight for item 3: 120 30

Enter knapsack capacity: 50

The maximum value that can be put in knapsack is 220