Implement 0/1 Knapsack problem using dynamic programming.

Code:

```
#include <stdio.h>

// Function to return maximum of two integers
int max(int a, int b) {
    return (a > b) ? a : b;
}

// Function to solve 0/1 Knapsack using Dynamic Programming
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];

    printf("Enter the values of the items:\n");
    for (int i = 0; i < n; i++) {
        printf("Value of item %d: ", i + 1);
        scanf("%d", &val[i]);
    }
```

```
    printf("Enter the weights of the items:\n");
    for (int i = 0; i < n; i++) {
        printf("Weight of item %d: ", i + 1);
        scanf("%d", &wt[i]);
    }

    printf("Enter the capacity of the knapsack: ");
    scanf("%d", &W);

    int result = knapsack(W, wt, val, n);
    printf("Maximum value in Knapsack = %d\n", result);

    return 0;
}
```

Output: