1. **Knapsack using dynamic programming**

**Code:**

#include<stdio.h>

// Function to find maximum of two integers

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

// Function to solve knapsack problem using dynamic programming

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

{

int i, w;

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

// Build table K[][] in bottom up manner

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 the number of items: ");

scanf("%d", &n);

int val[n], wt[n];

printf("Enter the values and weights of the items:\n");

for(int i = 0; i < n; i++) {

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

}

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

scanf("%d", &W);

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

return 0;

}

**Output:**

Enter the number of items: 4

Enter the values and weights of the items:

40

15

12

56

45

23

12

47

Enter the capacity of the knapsack: 150

The maximum value that can be put in a knapsack of capacity 150 is: 109

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

Process exited after 22.73 seconds with return value 0

Press any key to continue . . .

