**15. Write an algorithm and program to implement 0/1 Knapsack.**

// solution for 0-1 Knapsack problem

#include <stdio.h>

// A utility function that returns

// maximum of two integers

int max(int a, int b)

{

return (a > b) ? a : b;

}

// Returns the maximum value that

// can be put in a knapsack of capacity W

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];

}

// Driver Code

int main()

{

int val[] = { 60, 100, 120 };

int wt[] = { 10, 20, 30 };

int W = 50;

int n = sizeof(val) / sizeof(val[0]);

printf("%d", knapSack(W, wt, val, n));

return 0;

**}**

**OUTPUT**

