Name:

Roll No.:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Aim- Implementation of 0-1 knapsack problem using branch and bound approach.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class Zero\_One\_Knapsack

{

public void solve(int[] wt, int[] val, int W, int N)

{

int NEGATIVE\_INFINITY = Integer.MIN\_VALUE;

int[][] m = new int[N + 1][W + 1];

int[][] sol = new int[N + 1][W + 1];

for (int i = 1; i <= N; i++)

{

for (int j = 0; j <= W; j++)

{

int m1 = m[i - 1][j];

int m2 = NEGATIVE\_INFINITY;

if (j >= wt[i])

m2 = m[i - 1][j - wt[i]] + val[i];

m[i][j] = Math.max(m1, m2);

sol[i][j] = m2 > m1 ? 1 : 0;

}

}

int[] selected = new int[N + 1];

for (int n = N, w = W; n > 0; n--)

{

if (sol[n][w] != 0)

{

selected[n] = 1;

w = w - wt[n];

}

else

selected[n] = 0;

}

System.out.print("\nItems with weight ");

for (int i = 1; i < N + 1; i++)

if (selected[i] == 1)

System.out.print(val[i] +" ");

System.out.println("are selected by knapsack algorithm.");

}

public static void main (String[] args)

{

Scanner scan = new Scanner(System.in);

Zero\_One\_Knapsack ks = new Zero\_One\_Knapsack();

System.out.println("Enter number of elements ");

int n = scan.nextInt();

int[] wt = new int[n + 1];

int[] val = new int[n + 1];

System.out.println("Enter weight for "+ n +" elements");

for (int i = 1; i <= n; i++)

wt[i] = scan.nextInt();

System.out.println("Enter value for "+ n +" elements");

for (int i = 1; i <= n; i++)

val[i] = scan.nextInt();

System.out.println("Enter knapsack weight ");

int W = scan.nextInt();

ks.solve(wt, val, W, n);

scan.close();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

OUTPUT:

C:\Users\K!SH>cd\

C:\>javac Zero\_One\_Knapsack.java

C:\>java Zero\_One\_Knapsack

Enter number of elements

4

Enter weight for 4 elements

10 15 6 9

Enter value for 4 elements

2 5 8 1

Enter knapsack weight

30

Items with weight 5 8 1 are selected by knapsack algorithm.

\*/

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*