Assignment No. 2

Aim: Implement 0/1 knapsack using Dynamic Programming.

Theory:

The 0/1 Knapsack problem is a classic optimization problem that asks, given a set of items with weights and values, and a knapsack with a maximum weight capacity, what is the maximum value that can be obtained by packing items in the knapsack such that the total weight is less than or equal to the capacity. The Dynamic Programming approach to solve this problem involves creating at able with rows representing the items and columns representing the remaining capacity of the knapsack. The table is then filled out by considering each item and calculating the maximum value that can be obtained with or without the item at each remaining capacity of the knapsack. The final entry in the table gives the maximum value that can be obtained.

- Time Complexity:

O(nW),where n is number of items,W is maximum weight capacity of knapsack.

- Space Complexity: O(nW)

Code: -

```
public class Knapsack {

public static int knapsack(int[] values, int[] weights, int W) {

int n = values.length;

int[][] dp = new int[n + 1][W + 1];

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

for (int j = 1; j <= W; j++) {

if (weights[i - 1] <= j) {

dp[i][j] = Math.max(values[i - 1] + dp[i - 1][j -

weights[i - 1]], dp[i - 1][j]);

} else {

dp[i][j] = dp[i - 1][j];

}

}
```

```
}
return dp[n][W];
}
public static void main(String[] args) {
int[] values = {60, 100, 120};
int[] weights = {10, 20, 30};
int W = 50;
System.out.println("Maximum value that can be obtained = " +
knapsack(values, weights, W));
}
}
```

Output: -

```
Maximum value that can be obtained = 220
```