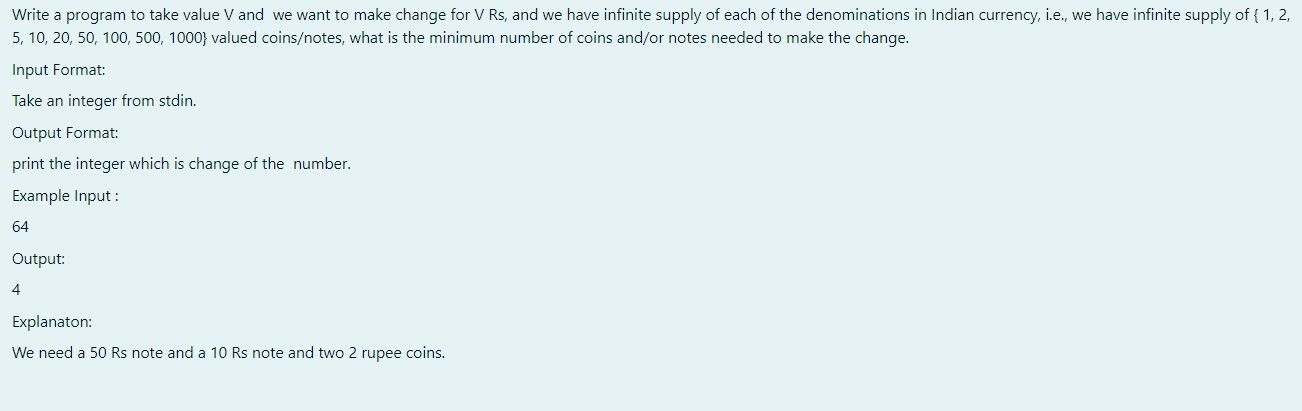
GREEDY ALGORITHM

**QUESTION 3.A AIM:**



**ALGORITHM:**

**Step 1: Start**

**Step 2:** Input the integer v, the amount for which denominations are needed.

**Step 3:** Initialize an array denominations with values {1000, 500, 100, 50, 20, 10,

5, 2, 1}.

**Step 4:** Initialize count to 0 to keep track of the total number of denominations.

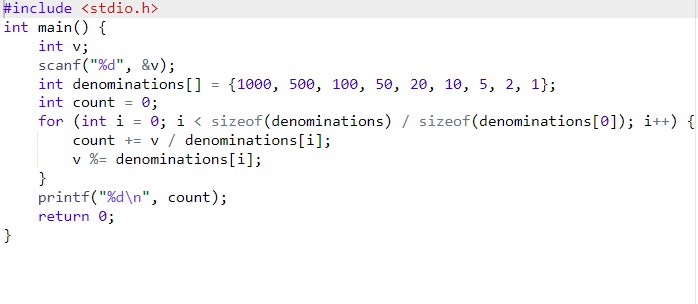
**Step 5:** For each denomination in denominations:

* Divide v by the current denomination to find how many of that denomination are needed and add the result to count.
* Update v to the remainder after division.

**Step 6:** Print the value of count.

**Step 7: Stop**

**PROGRAM :**



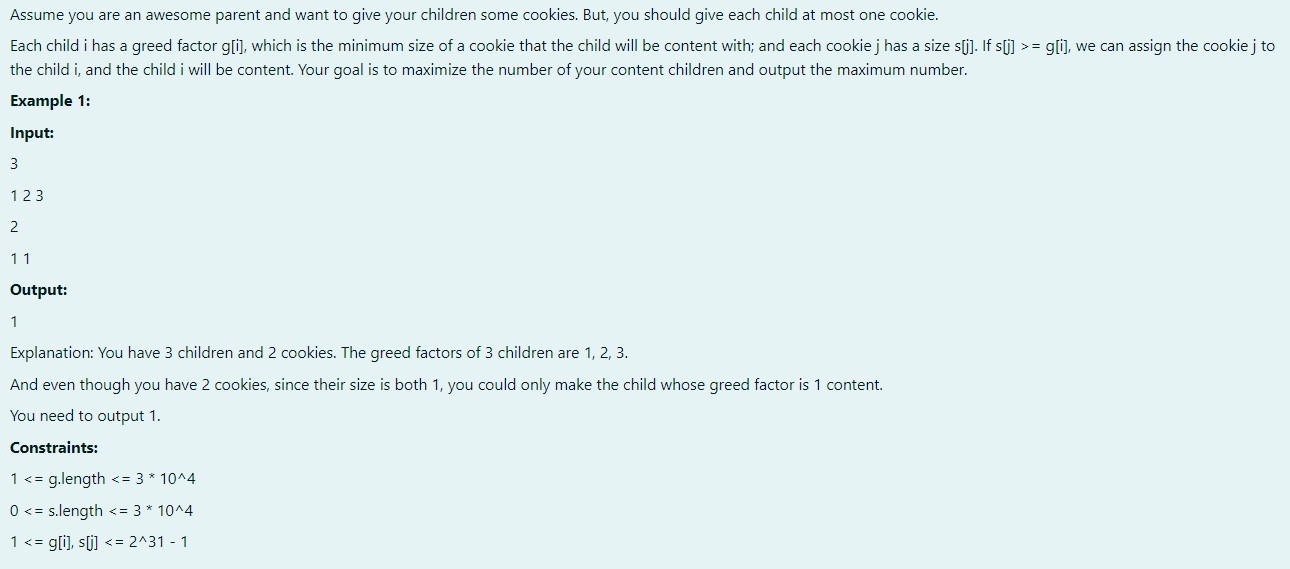
**OUTPUT:**



**RESULT :**

The above program is executed successfully.

**QUESTION 3.B AIM:**



**ALGORITHM:**

**Step 1: Start**

**Step 2:** Input the integer n, the number of elements in array g.

**Step 3:** Input n integers into array g.

**Step 4:** Input the integer m, the number of elements in array c.

**Step 5:** Input m integers into array c.

**Step 6:** Initialize co to 0 to count compatible pairs.

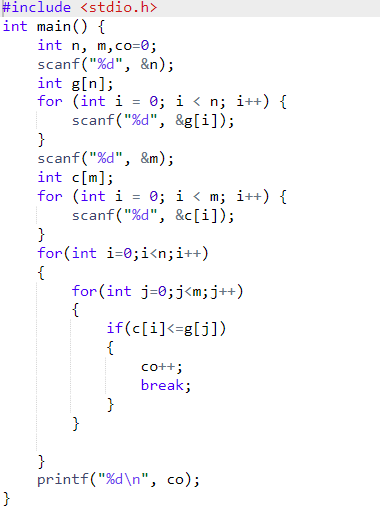
**Step 7:** For each element in g, check if there exists an element in c such that c[i] <= g[j]:

* If a compatible element is found, increment co and stop checking further for that g[j].

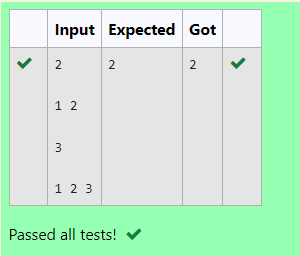
**Step 8:** Print the value of co.

**Step 9: Stop**

**PROGRAM :**



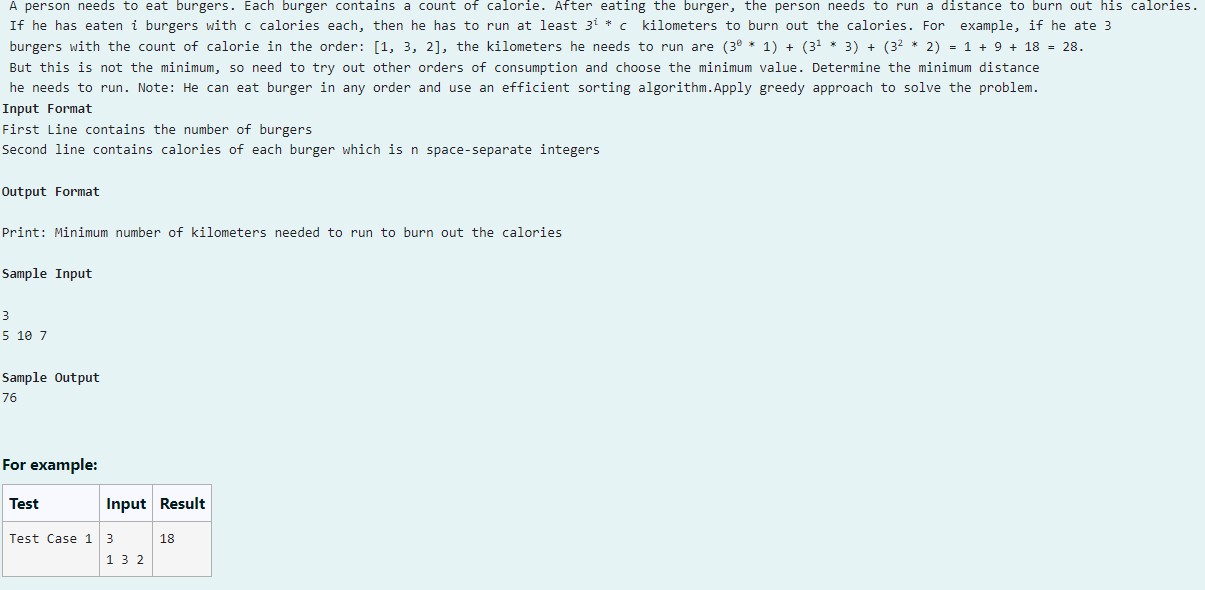
**OUTPUT:**



**RESULT :**

The above program is executed successfully.

**QUESTION 3.C AIM:**



**ALGORITHM:**

**Step 1: Start**

**Step 2:** Input the integer n, the number of elements in array c.

**Step 3:** Input n integers into array c.

**Step 4:** Sort the array c in descending order.

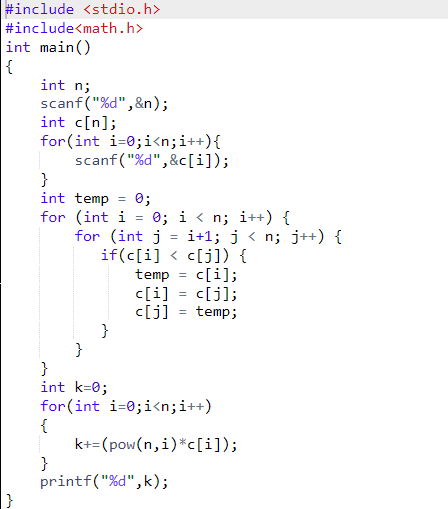
**Step 5:** Initialize k to 0 to store the weighted sum.

**Step 6:** For each element c[i], calculate c[i] \* n^i and add it to k.

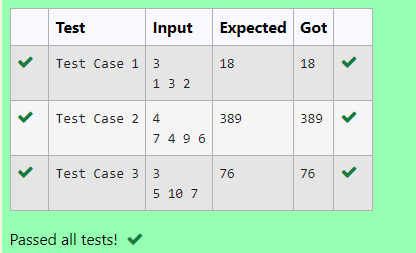
**Step 7:** Print the value of k.

**Step 8: Stop**

**PROGRAM:**



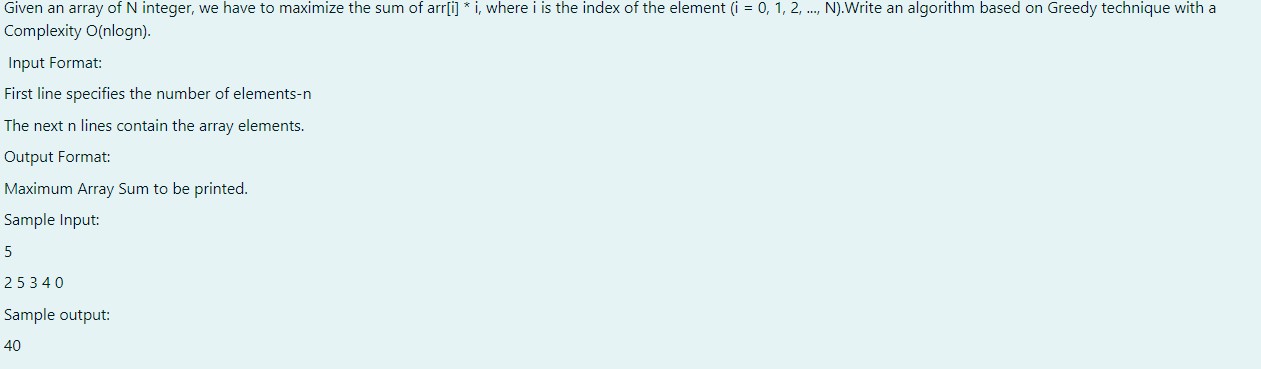
**OUTPUT:**



**RESULT :**

The above program is executed successfully.

**QUESTION 3.D AIM:**



**ALGORITHM:**

**Step 1: Start**

**Step 2:** Input the integer n, the number of elements in array a.

**Step 3:** Input n integers into array a.

**Step 4:** Sort the array a in ascending order.

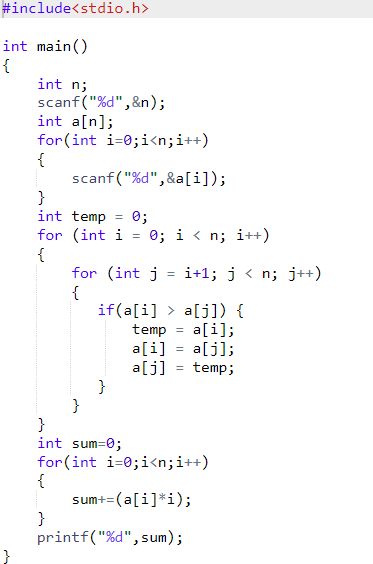
**Step 5:** Initialize sum to 0 to store the weighted sum.

**Step 6:** For each element a[i], multiply it by its index i and add it to sum.

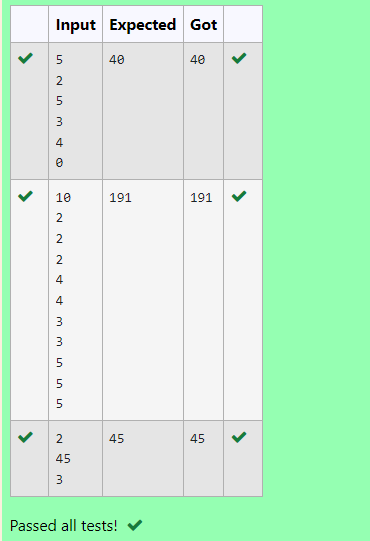
**Step 7:** Print the value of sum.

**Step 8: Stop**

**PROGRAM:**

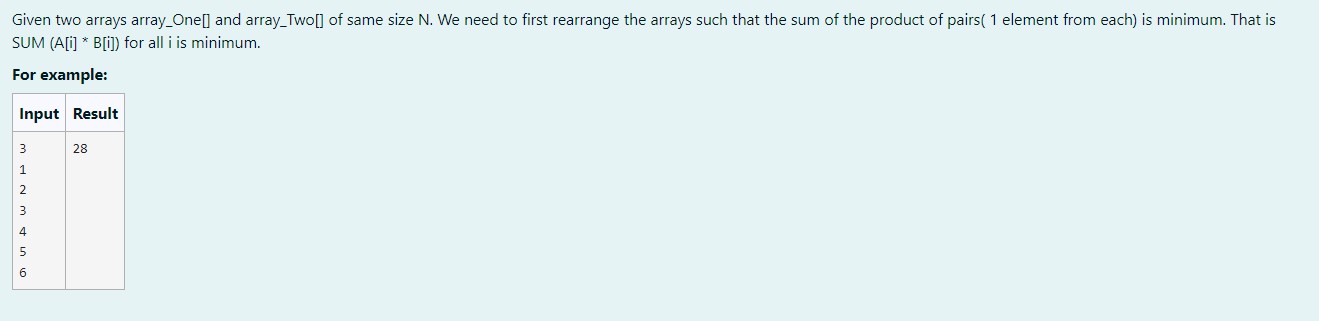


**OUTPUT:**



**QUESTION 3.E**

**AIM:**



**ALGORITHM :**

**Step 1: Start**

**Step 2:** Input the integer n, the number of elements in arrays a and b.

**Step 3:** Input n integers into array a. **Step 4:** Input n integers into array b. **Step 5:** Sort array a in ascending order. **Step 6:** Sort array b in descending order.

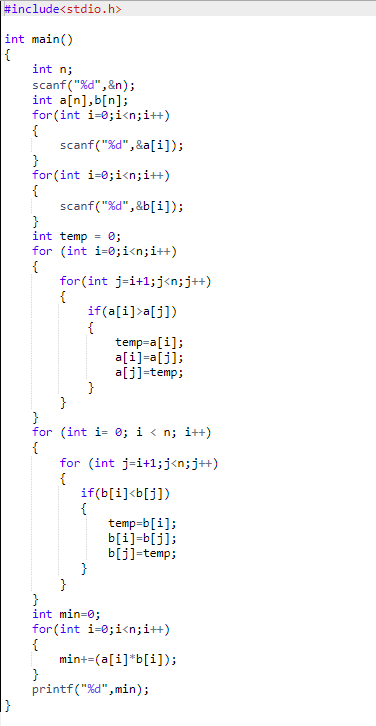
**Step 7:** Initialize min to 0 to store the minimum weighted sum.

**Step 8:** For each index i, multiply a[i] and b[i] and add the result to min.

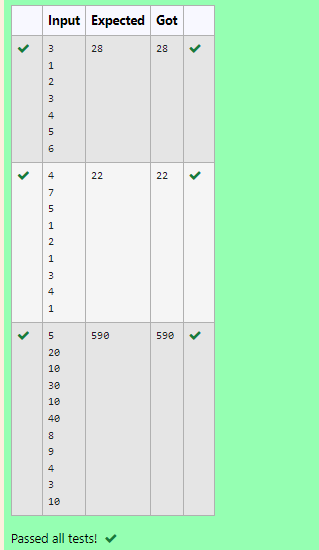
**Step 9:** Print the value of min.

**Step 10: Stop**

**PROGRAM :**



**OUTPUT:**



**RESULT:**

The above program is executed successfully.