NAME: SREYA G **REGISTER NO.:** 230701334

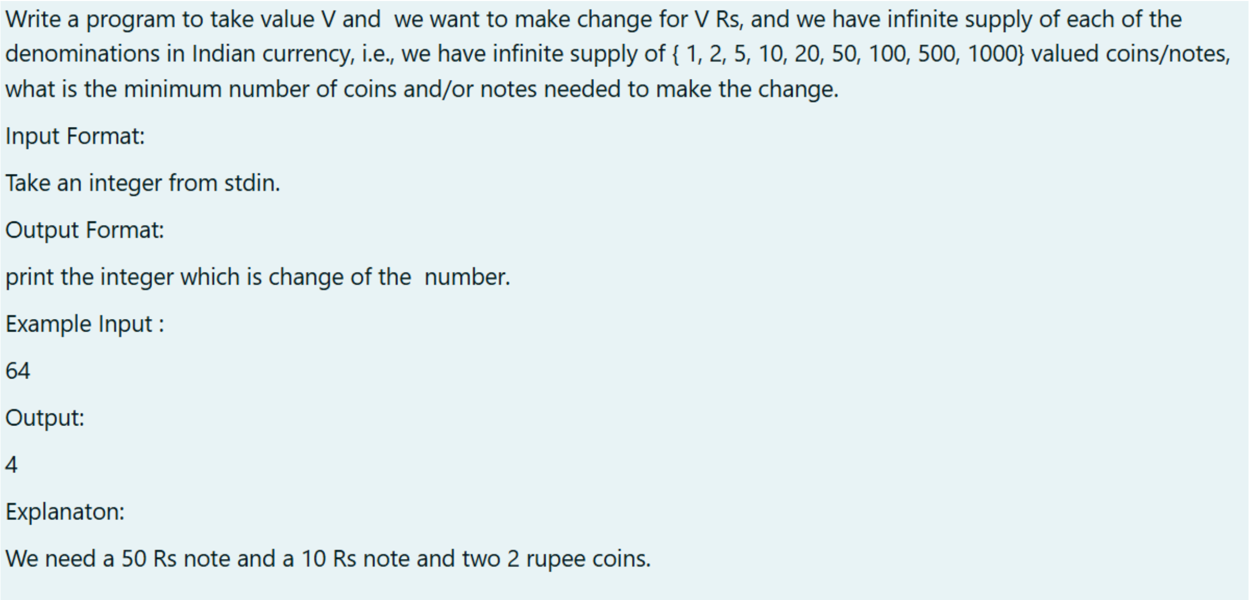
CLASS : CSE F **DATE:** 22/08/2024

EX – 3:

GREEDY ALGORITHMS:

PROBLEM 1:

AIM:



Write a program to take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.

Input Format:

Take an integer from stdin.

Output Format:

print the integer which is change of the number.

Example Input :

64

Output:

4

Explanaton:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

ALGORITHM:

1. Input V: Read the value .

2. Initialize denominations: Use an array of currency denominations in descending order.

3. Initialize count : Set count to zero.

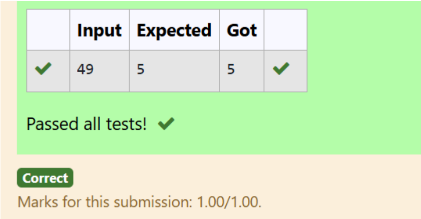
4. Iterate through denominations:

```
  - For each denomination:
    - Add  V / denomination to count.
    - Update V to V % denomination.
5. Output count : Print the total count of notes/coins.
6. End.
```

CODE:

```c
#include<stdio.h>
int main()
{
   int V;
   scanf("%d",&V);
   int denominations[] = {1000,500,100,50,20,10,5,2,1};
   int count = 0;
   for(int i = 0;i < sizeof(denominations) / sizeof(denominations[0]);i++)
   {
      count += V/denominations[i];
      V %= denominations[i];
   }
   printf("%d\n",count);
   return 0;
}
```

OUTPUT:



	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

Correct

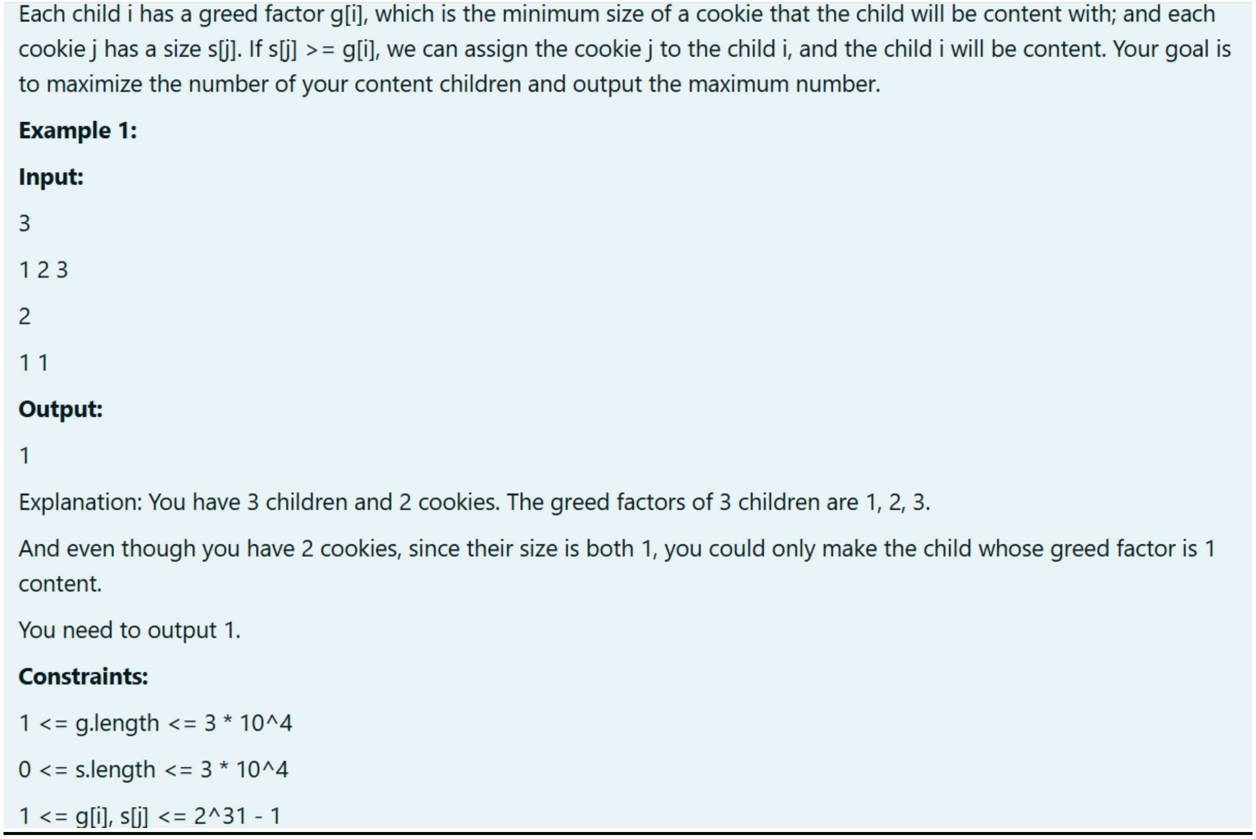
Marks for this submission: 1.00/1.00.

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 2:

AIM:



Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.

Example 1:

Input:

3

1 2 3

2

1 1

Output:

1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to output 1.

Constraints:

1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1

ALGORITHM:

1. Input n and array g.

2. Input m and array c.

3. Initialize co = 0.

4. For each c[i], check if c[i] <= g[j] for any g[j]. If true, increment co and break.

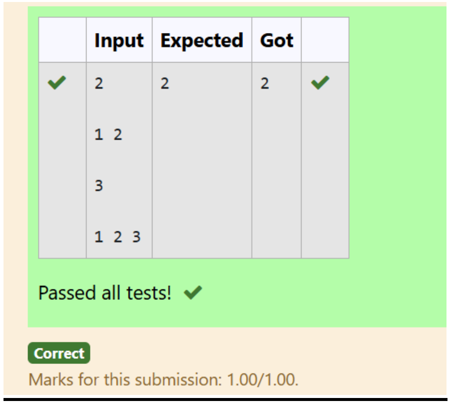
5. Output co.

6. End.

CODE:

```
#include<stdio.h>
int main()
{
    int n,m;
    int co = 0;
    scanf("%d",&n);
    int g[n];
    for(int i = 0;i < n;i++)
    {
        scanf("%d",&g[i]);
    }
    scanf("%d",&m);
    int c[m];
    for(int i = 0;i < m;i++)
    {
        scanf("%d",&c[i]);
    }
    for(int i = 0;i < n;i++)
    {
        for(int j = 0;j < m;j++)
        {
            if(c[i] <= g[j])
            {
                co++;
                break;
            }
        }
    }
    printf("%d\n",co);
}
```

OUTPUT:



	Input	Expected	Got	
✓	2 1 2 3 1 2 3	2	2	✓

Passed all tests! ✓

Correct

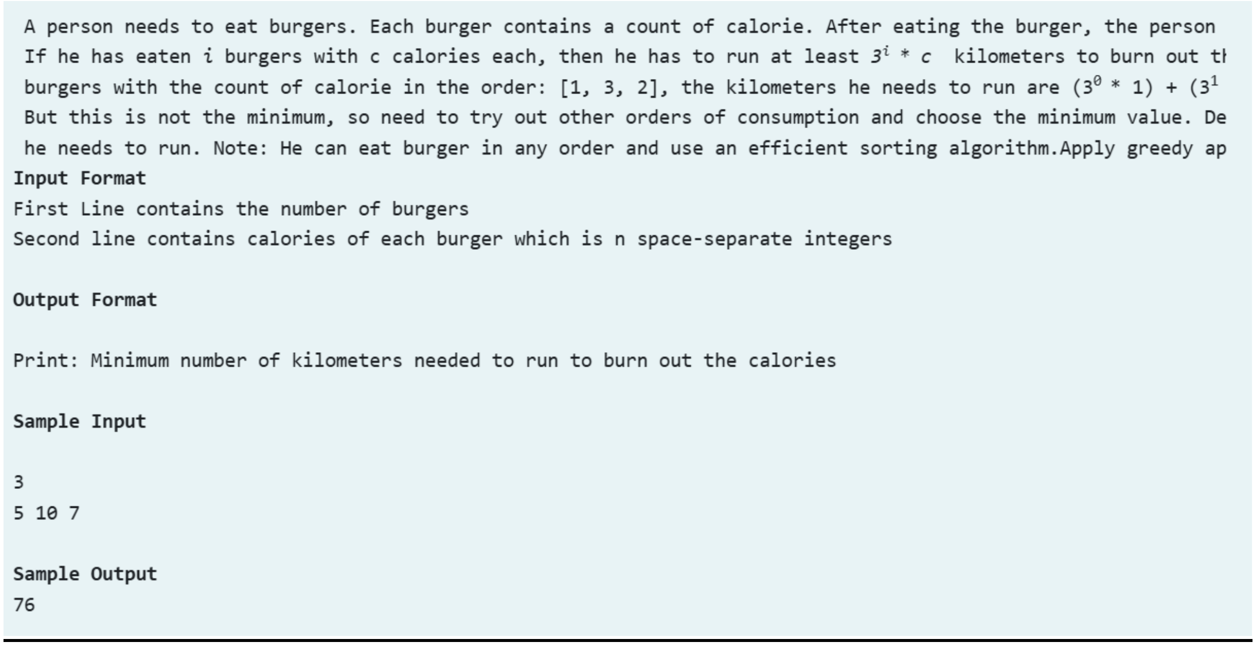
Marks for this submission: 1.00/1.00.

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 3:

AIM:



A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person
If he has eaten *i* burgers with c calories each, then he has to run at least $3^{i} * c$ kilometers to burn out th
burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^{0} * 1)$ + $(3^{1}$
But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. De
he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy ap

Input Format

First Line contains the number of burgers
Second line contains calories of each burger which is n space-separate integers

Output Format

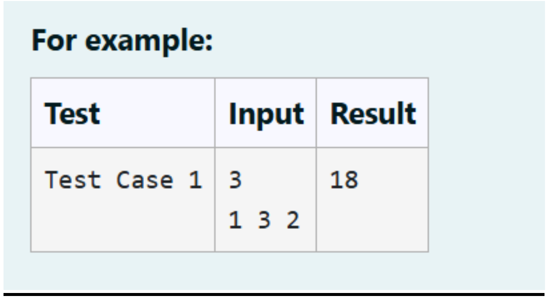
Print: Minimum number of kilometers needed to run to burn out the calories

Sample Input

```
3
5 10 7
```

Sample Output

```
76
```



For example:

Test	Input	Result
Test Case 1	3 1 3 2	18

ALGORITHM:

1. Input n and array a.

2. Sort a in descending order using Bubble Sort.

3. Initialize km = 0.

4. Calculate km:

   - For each element a[i], add a[i] * (n^i) to km.

5. Output km.

6. End.

CODE:

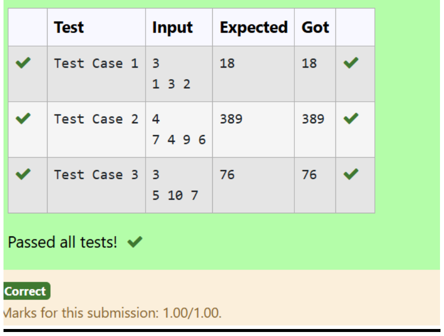
```
#include<stdio.h>
```

```
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i = 0;i < n;i++)
    {
        scanf("%d",&a[i]);
    }
    int km = 0;

    for(int i = 0;i < n-1;i++)
    {
        for(int j = 0;j < n-i-1;j++)
        {
            if(a[j] < a[j+1])
            {
                int t = a[j];
                a[j] = a[j+1];
                a[j+1] = t;
            }
        }
    }
    for(int i = 0;i < n;i++)
    {
        int p = 1;
        if(i == 0)
        km += (p*a[0]);
        else
        {
            for(int j = 1;j <= i;j++)
            {
                p *= n;
            }
            km += (p * a[i]);
        }

    }
    printf("%d",km);
}
```

OUTPUT:



	Test	Input	Expected	Got	
✓	Test Case 1	3 1 3 2	18	18	✓
✓	Test Case 2	4 7 4 9 6	389	389	✓
✓	Test Case 3	3 5 10 7	76	76	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

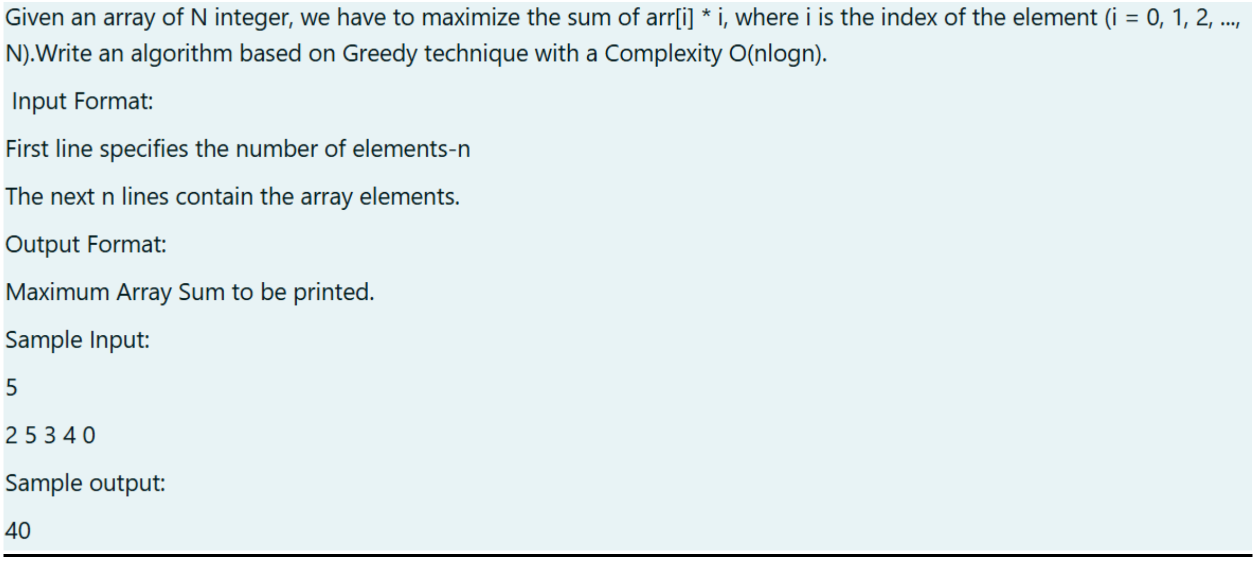
RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 4:

AIM;

{



Given an array of N integer, we have to maximize the sum of arr[i] * i, where i is the index of the element (i = 0, 1, 2, ..., N).Write an algorithm based on Greedy technique with a Complexity O(nlogn).

Input Format:

First line specifies the number of elements-n

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

5

2 5 3 4 0

Sample output:

40

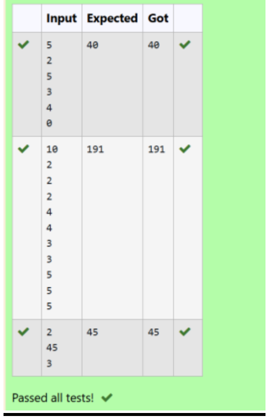
ALGORITHM:

```
1. Input n and array arr.
2. Sort arr in ascending order.
3. Initialize sum = 0.
4. Compute sum += arr[i] * i for each element.
5. Output sum.
6. End.
```

CODE:

```
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int arr[n];
    for(int i = 0;i < n;i++)
    {
        scanf("%d",&arr[i]);
    }
    for(int i = 0;i < n;++i)
    {
        for(int j = i+1;j < n;++j)
        {
            if(arr[i] > arr[j])
            {
                int a = arr[i];
                arr[i] = arr[j];
                arr[j] = a;
            }
        }
    }
    int sum = 0;
    for(int i = 0;i < n;i++)
    {
        sum += arr[i]*i;
    }
    printf("%d",sum);
}
```

OUTPUT:



	Input	Expected	Got	
✓	5 2 5 3 4 0	40	40	✓
✓	10 2 2 2 4 4 3 3 5 5 5	191	191	✓
✓	2 45 3	45	45	✓

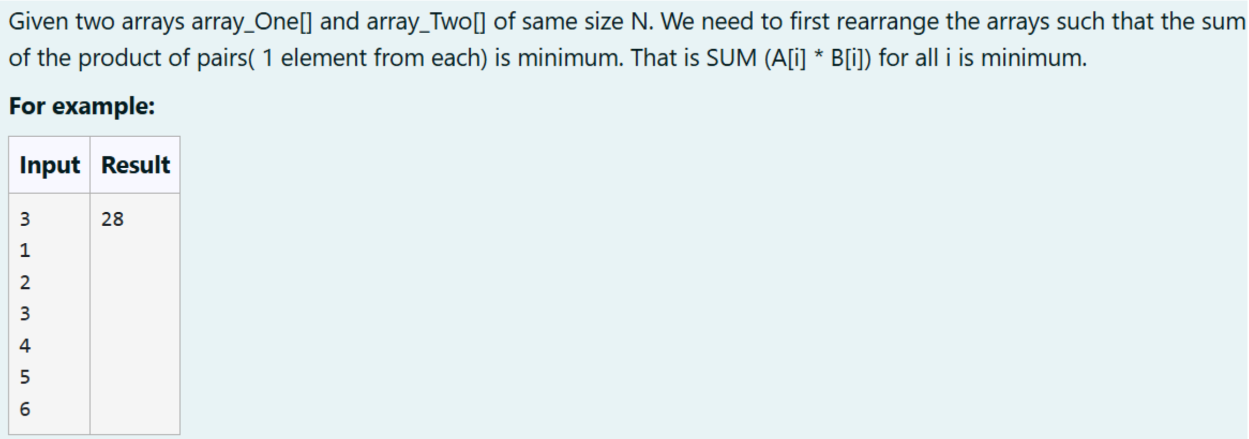
Passed all tests! ✓

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 5:

AIM:



Given two arrays array_One[] and array_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs(1 element from each) is minimum. That is SUM (A[i] * B[i]) for all i is minimum.

For example:

Input	Result
3 1 2 3 4 5 6	28

ALGORITHM:

1. Input N and arrays A and B.

2. Sort A in ascending order using Bubble Sort.

3. Sort B in descending order using Bubble Sort.

4. Initialize s = 0.

5. Compute the sum: Add A[i] * B[i] to s for all elements.

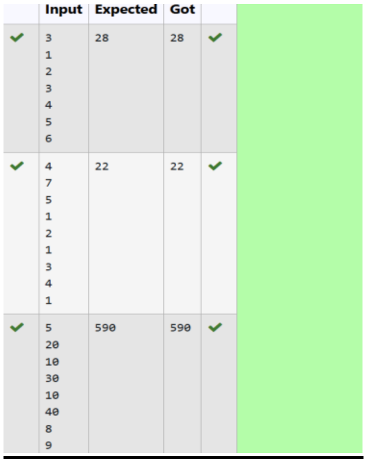
6. Output s.

7. End.

CODE:

```
#include<stdio.h>
int main()
{
    int N;
    scanf("%d",&N);
    int A[N],B[N];
    for(int i = 0;i < N;i++)
    {
        scanf("%d",&A[i]);
    }
    for(int i = 0;i < N;i++)
    {
        scanf("%d",&B[i]);
    }
    for(int i = 0;i < N-1;i++)
    {
        for(int j = 0;j < N-i-1;j++)
        {
            if(A[j] > A[j+1])
            {
                int t = A[j];
                A[j] = A[j+1];
                A[j+1] = t;
            }
            if(B[j] < B[j+1])
            {
                int t = B[j];
                B[j] = B[j+1];
                B[j+1] = t;
            }
        }
    }
    int s = 0;
    for(int i = 0;i < N;i++)
    {
        s += (A[i] * B[i]);
    }
    printf("%d",s);
}
```

OUTPUT:



	Input	Expected	Got	
✓	3 1 2 3 4 5 6	28	28	✓
✓	4 7 5 1 2 1 3 4 1	22	22	✓
✓	5 20 10 30 10 40 8 9	590	590	✓

RESULT:

Thus the code is executed successfully and gives the expected output.