Ex. No: 3 **Date: 26.08.24**

Register No.: 230701368 **Name: AL UMA**

____________GREEDY ALGOTITHM

3.A 1-G Coin Problem

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:

```
function calculate(v):
    set c = 0
```

```
while v / 1000 != 0:
    increment c by 1
    decrement v by 1000

while v / 500 != 0:
    increment c by 1
    decrement v by 500

while v / 100 != 0:
    increment c by 1
    set v = v / 100

while v / 50 != 0:
    increment c by 1
    set v = v / 50

while v / 20 != 0:
    increment c by 1
    decrement v by 20

while v / 10 != 0:
    increment c by 1
    decrement v by 10

while v / 5 != 0:
    increment c by 1
    decrement v by 5

while v / 2 != 0:
    increment c by 1
    decrement v by 2

while v / 1 != 0:
    increment c by 1
    decrement v by 1

return c
```

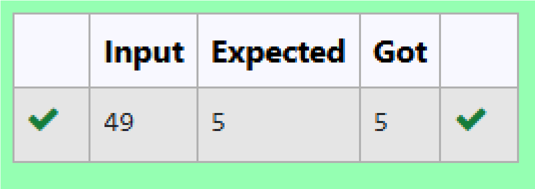
PROGRAM:

```
#include<stdio.h>
int main()
{
int v;
scanf("%d",&v);
int c=0;
while(v/1000 !=0)
{
c+=1;
v=v-1000;
}
while(v/500 !=0)
{
c+=1;
v=v-500;
}
while(v/100!=0)
{
c+=1;
v=v/100;
```

```
}
while(v/50!=0)
{
c+=1;
v=v/50;
}
while(v/20!=0)
{
c+=1;
v=v-20;
}
while(v/10!=0)
{
c+=1;
v=v-10;
}
while(v/5!=0)
{
c+=1;
v=v-5;
}
while(v/2!=0)
{
c+=1;
v=v-2;
}
while(v/1!=0)
{
c+=1;
v=v-1;
```

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

OUTPUT:



	Input	Expected	Got	
✓	49	5	5	✓

3.B 2-G Cookies Problem

AIM:

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

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

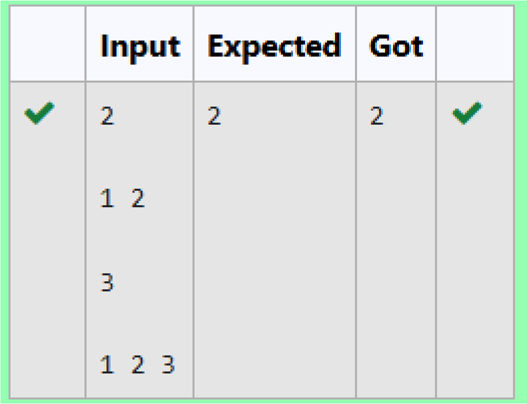
ALGORITHM:

```
function calculate(n, n1, a, b):
    set c = 0

    for i = 0 to n - 1:
        for j = 0 to n1 - 1:
            if a[i] >= b[j]:
                increment c by 1
                break

    return c
```

PROGRAM:

```
#include<stdio.h>
int main()
{
    int n,n1;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    scanf("%d",&n1);
    int b[n1];
    for(int i=0;i<n1;i++)
    {
        scanf("%d",&b[i]);
    }
    int c=0;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(a[i]>=b[j])
            {
                c+=1;
                break;
            }
        }
    }
    printf("%d",c);
}
```

OUTPUT:



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

3.C 3-G Burger Problem

AIM:

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories. If he has eaten *i* burgers with c calories each, then he has to run at least $3^i * c$ kilometers to burn out the calories. For example, if he ate 3 burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$.
But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy approach to solve the problem.

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

ALGORITHM

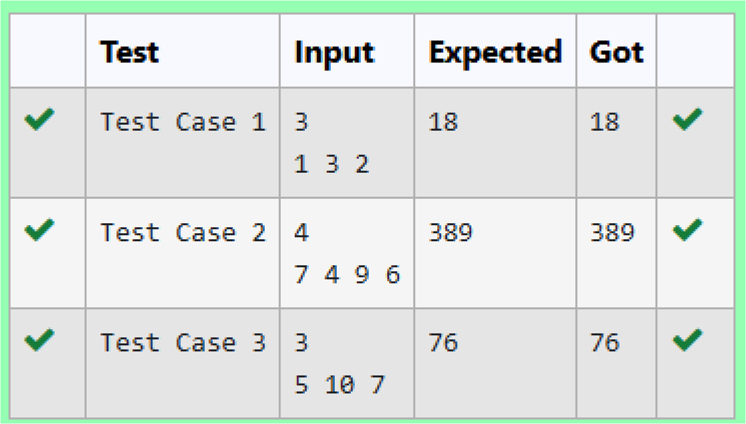
```
function calculate(n, a):
    set km = 0


    for i = 0 to n-1:
        for j = 0 to n-i-2:
            if a[j] < a[j+1]:
                swap a[j] and a[j+1]
```

```
for i = 0 to n-1:
    set p = 1
    if i == 0:
        increment km by (p * a[0])
    else:
        for j = 1 to i:
            multiply p by n
        increment km by (p * a[i])

return km
```

PROGRAM:

```
#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 temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }

    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	✓

4.D 4-G Array Sum Max Problem

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

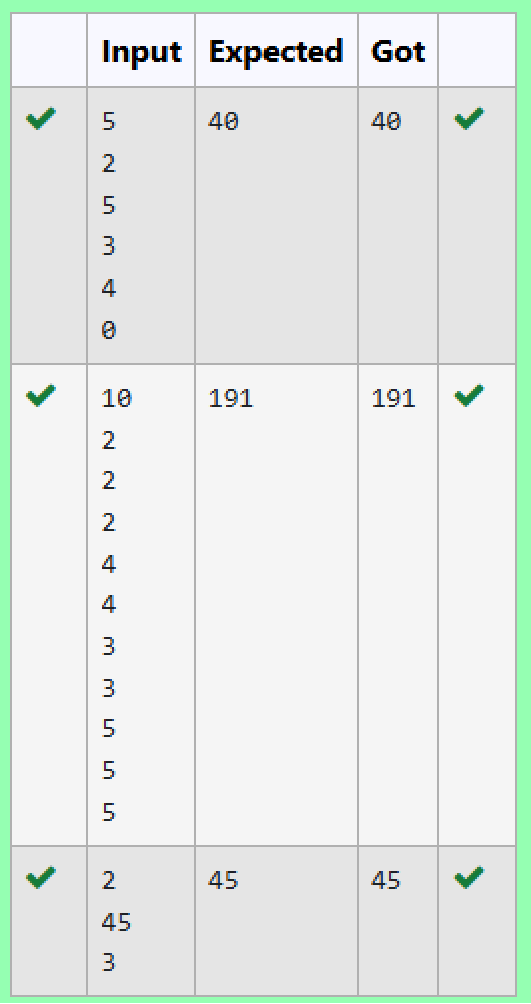
ALGORITHM:

```
function calculate(n, a):
    for i = 0 to n-1:
        read a[i]

    for i = 0 to n-2:
        for j = 0 to n-i-2:
            if a[j] > a[j+1]:
                swap a[j] and a[j+1]

    set s = 0
    for i = 0 to n-1:
        increment s by (a[i] * i)

    return s
```

PROGRAM

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

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	✓

3.E 5-G Product of Array elements Minimum

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.

ALGORITHM:

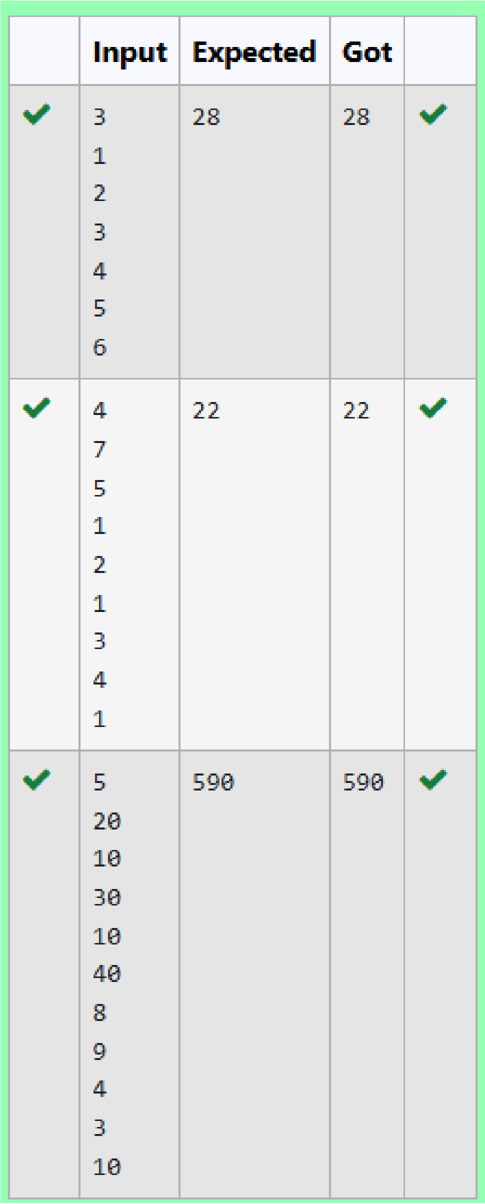
```
Function Main()
   // Step 1: Read the number of elements
   Initialize n          // Number of elements
   Read n from user      // Read the input value for n

   // Step 2: Initialize the arrays
   Initialize array_One of size n  // Array to hold the first set of values
   Initialize array_Two of size n  // Array to hold the second set of values

   // Step 3: Input elements for array_One
   For i from 0 to n-1  // Loop through each element index of array_One
      Read array_One[i] from user  // Read value into array_One at index i
   End For

   // Step 4: Input elements for array_Two
   For i from 0 to n-1  // Loop through each element index of array_Two
      Read array_Two[i] from user  // Read value into array_Two at index i
   End For

   // Step 5: Sort both arrays
   For i from 0 to n-2   // Outer loop for sorting (n-1 iterations)
```

```
        For j from 0 to n-i-2   // Inner loop for comparing adjacent elements (n-i-1 iterations)

            // Step 5.1: Sort array_One in ascending order
            If array_One[j+1] is less than array_One[j]
                // Swap elements in array_One
                Initialize temp as array_One[j]
                array_One[j] = array_One[j+1]
                array_One[j+1] = temp
            End If

            // Step 5.2: Sort array_Two in descending order
            If array_Two[j+1] is greater than array_Two[j]
                // Swap elements in array_Two
                Initialize temp as array_Two[j]
                array_Two[j] = array_Two[j+1]
                array_Two[j+1] = temp
            End If

        End For
    End For

    // Step 6: Initialize sum to 0
    Initialize sum as 0    // Variable to accumulate the sum of products

    // Step 7: Calculate the sum of products of corresponding elements
    For i from 0 to n-1   // Loop through each index of the arrays
        sum = sum + (array_One[i] * array_Two[i])  // Add the product of corresponding elements to sum
    End For
    // Step 8: Output the result
    Print sum   // Output the final sum of the products

End Function
```

PROGRAM

```
#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 temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
            if(b[j]<b[j+1])
            {
                int temp=b[j];
                b[j]=b[j+1];
                b[j+1]=temp;

            }
        }
    }
    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 4 3 10	590	590	✓