NAME: Venkateswar L

ROLL NUMBER: 230701376

SECTION: CSE-F

Design and Analysis Of Algorithms
CS23331

WEEK 3: GREEDY ALGORITHMS

PROGRAM 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.

ALGORITHM:

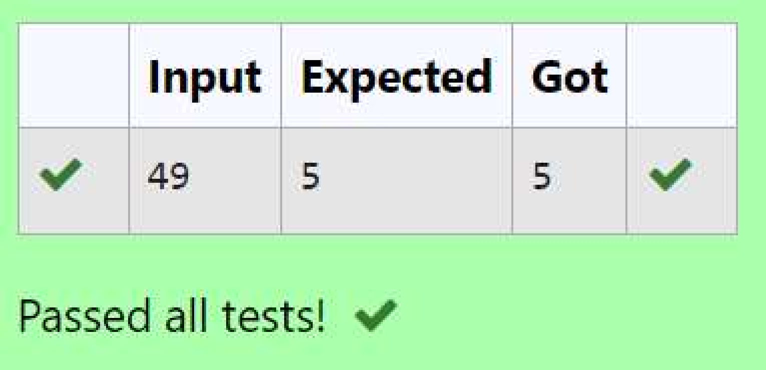
```
Step 1: Initialize all the variables required
Step 2: Define an array den[] and then take an input
Step 3: Iterate through the array and calculate c+=d/den[i] if den[i]<d
Step 4: Display C
```

PROGRAM:

```
#include<stdio.h>

int main()
{
   int d,c=0;
   scanf("%d",&d);

   int den[]={1000,500,100,50,20,10,5,2,1};
   int i=0;
   while(den[i]>d)
   {
      i++;
   }
   while(d!=0)
   {
      if (den[i]<d)
      {
         c+=d/den[i];
         d=d%den[i];
      }
      i++;
```

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

OUTPUT:



	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

RESULT: Thus the program executed successfully.

PROGRAM 2:

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.

ALGORITHM:

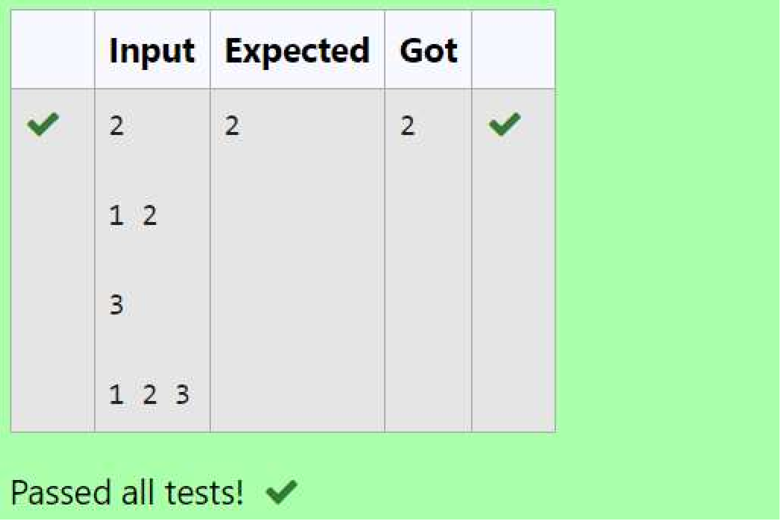
Step 1: Input the size of the first array g[] and its elements.
Step 2: Input the size of the second array s[] and its elements.
Step 3: Compare each element of g[] with the elements of s[].
Step 4: Output the result.

PROGRAM:

```
#include<stdio.h>

int main()
{
   int n;
   scanf("%d",&n);
   int g[n];
   for (int i=0;i<n;i++)
   {
      scanf("%d",&g[i]);
   }
   int c,r=0;
   scanf("%d",&c);
   int s[c];
   for(int j=0;j<c;j++)
   {
      scanf("%d",&s[j]);
   }
   for (int i=0;i<n;i++)
   {
      for (int j=0;j<c;j++)
      {
         if (s[j]>g[i])
         {
            r++;
```

```
                break;
            }
        }
    }
    printf("%d",r);
}
```

OUTPUT:



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

Passed all tests! ✓

RESULT: Thus the program was executed successfully.

PROGRAM 3:

AIM: A person needs to eat burgers. Each burger contains a count of calories. 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 *3i* * *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 (30 * 1) + (31 * 3) + (32 * 2) = 1 + 9 + 18 = 28.
But this is not the minimum, so I need to try out other orders of consumption and choose the minimum value. Determine the minimum distance He needs to run.

ALGORITHM:

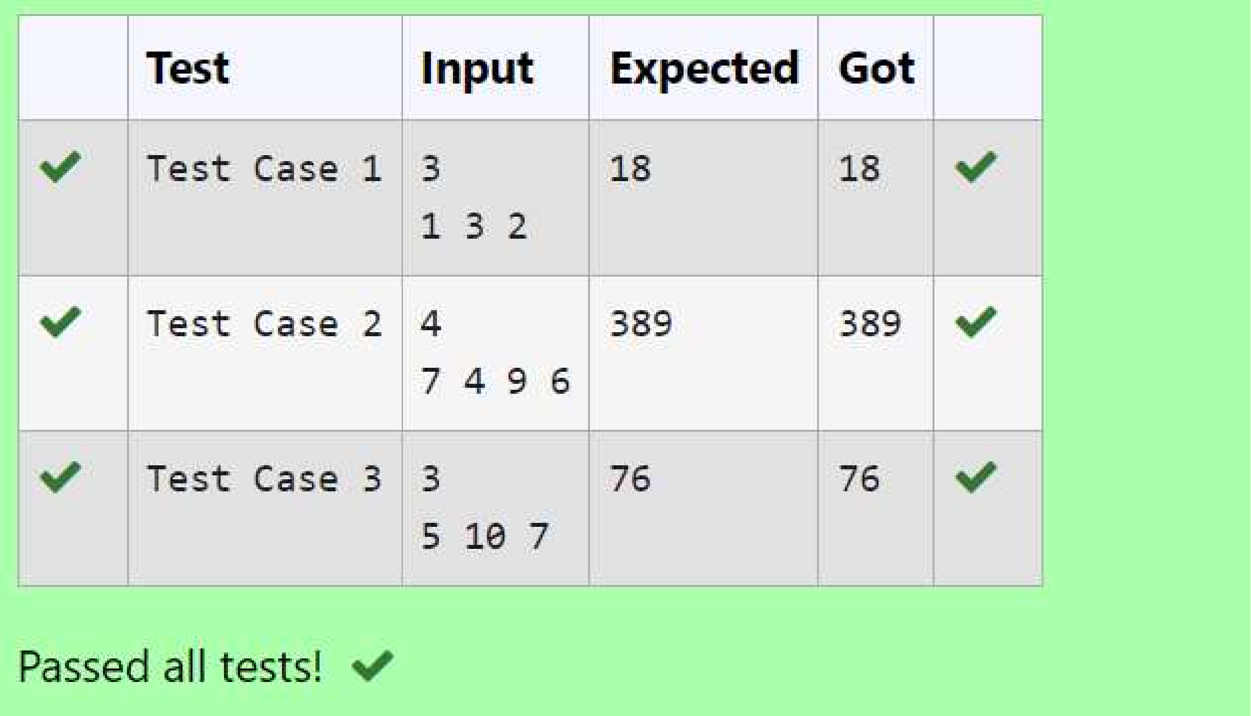
Step 1: Input the size of the array a[ ] and its elements.
Step 2: Sort the array in descending order.
Step 3: Calculate the sum with weighted powers.
Step 4: Output the result.

PROGRAM:

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

int compare(const void* a, const void* b)
{
   return (*(int*)b-*(int*)a);
}

int main()
{
   int n,sum=0;
   scanf("%d",&n);
   int a[n];

   for (int i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   qsort(a,n,sizeof(int),compare);

   for(int i=0;i<n;i++)
   {
      sum+=pow(n,i)*a[i];
```

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

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! ✓

RESULT: Thus the program was executed successfully.

PROGRAM 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).

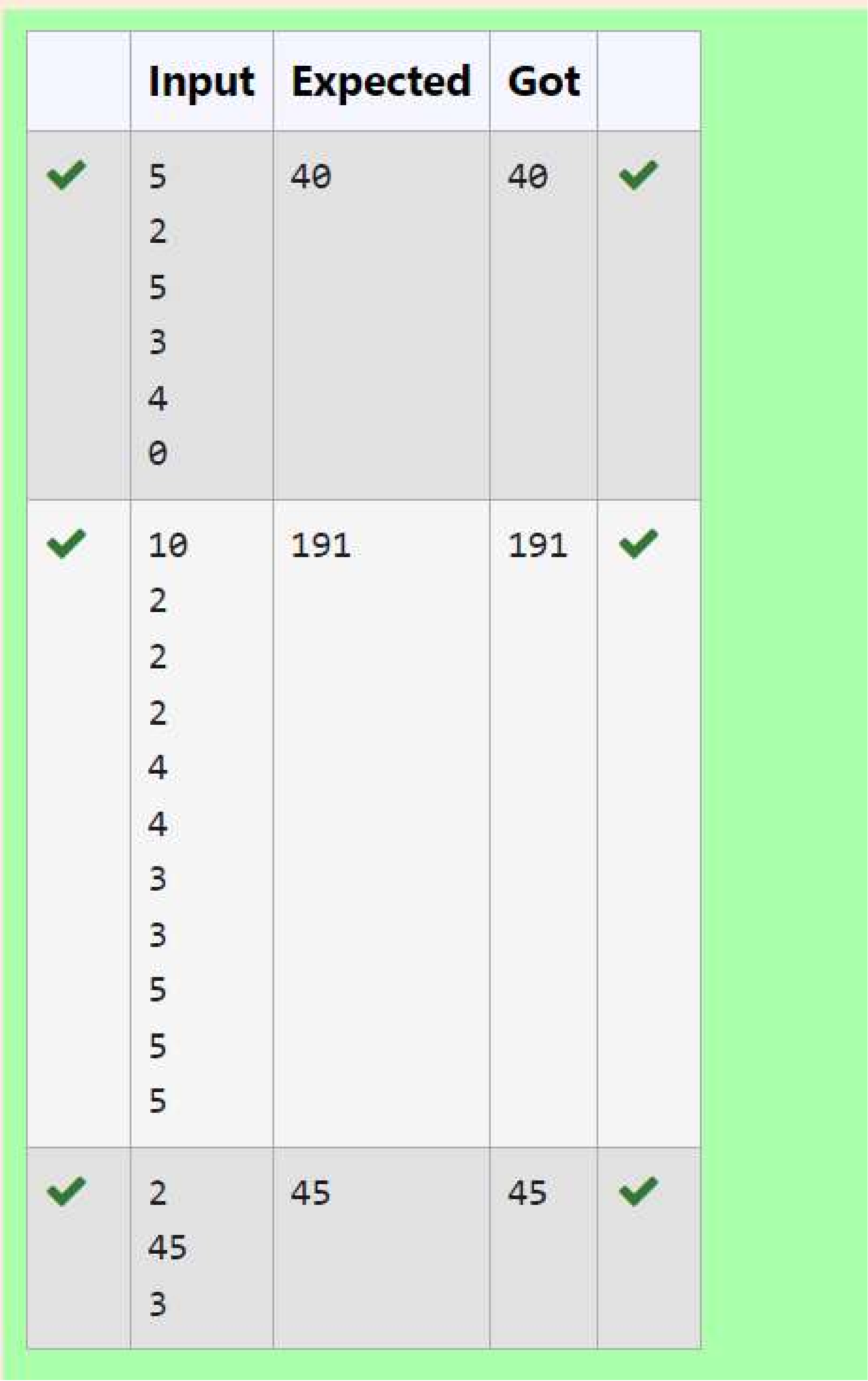
ALGORITHM:

```
Step 1: Input the size of the array a [ ] and its elements.
Step 2: Sort the array a [ ] in ascending order.
Step 3: Calculate the weighted sum.
Step 4: Output the result.
```

PROGRAM:

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

int compare(const void* a,const void* b)
{
   return (*(int*)a-*(int*)b);
}

int main()
{
   int n,sum=0;
   scanf("%d",&n);
   int a[n];
   for (int i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   qsort(a,n,sizeof(int),compare);

   for (int j=0;j<n;j++)
   {
      sum+=a[j]*j;
   }
   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	✓

RESULT: Thus the program executed successfully.

PROGRAM 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.

ALGORITHM:

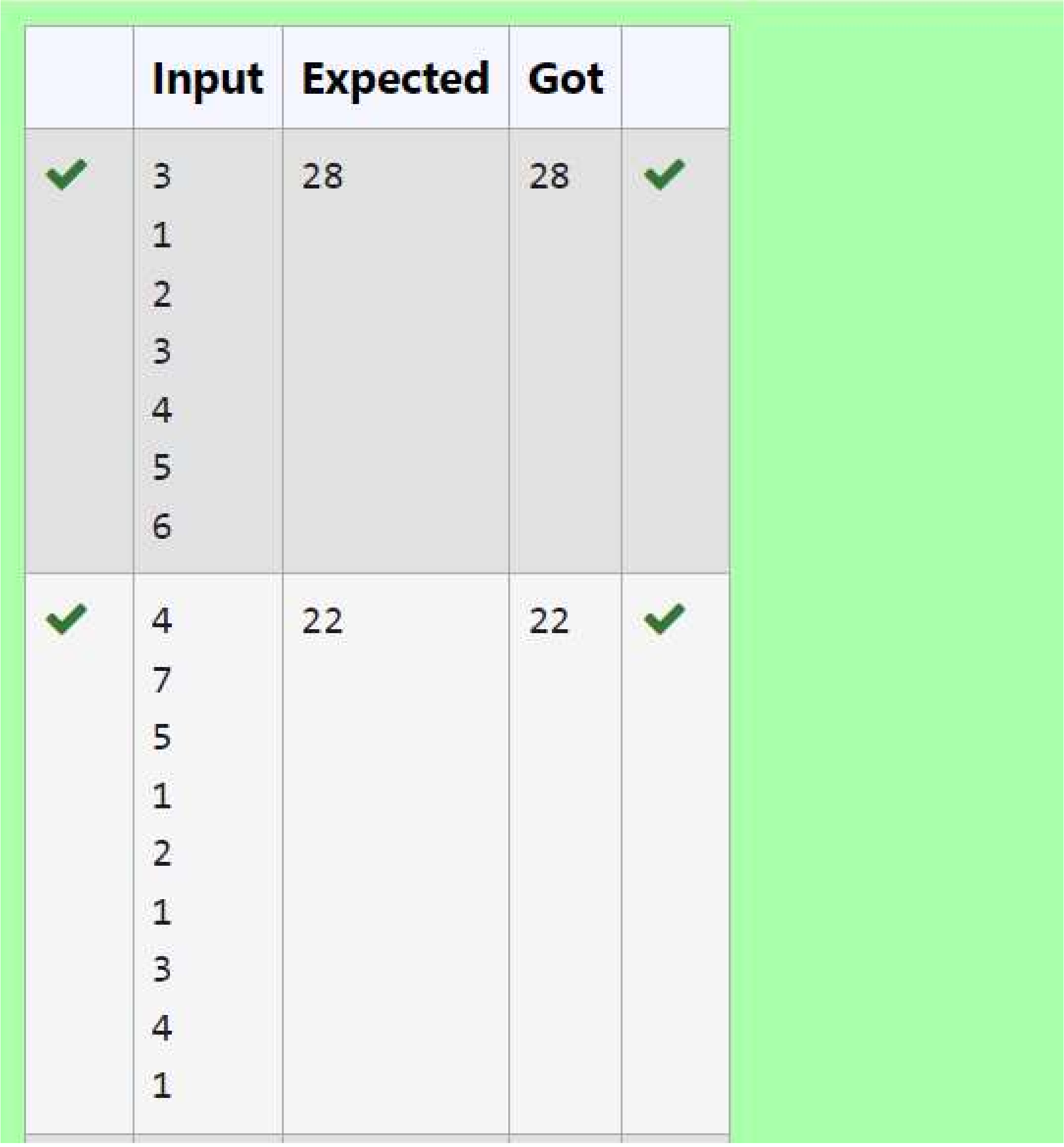
Step 1: Input the size of the arrays and the elements of both arrays a[] and b[].
Step 2: Sort array a[] in descending order and array b[] in ascending order.
Step 3: Calculate the sum of products.
Step 4: Output the result.

PROGRAM:

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

int compare(const void* a,const void* b)
{
   return (*(int*)a-*(int*)b);
}

int compare1(const void* a,const void* b)
{
   return (*(int*)b-*(int*)a);
}

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

   qsort(b,n,sizeof(int),compare);
```

```
    qsort(a,n,sizeof(int),compare1);

    for(int k=0;k<n;k++)
    {
        sum+=a[k]*b[k];
    }
    printf("%d",sum);
}
```

OUTPUT:



	Input	Expected	Got	
✓	3 1 2 3 4 5 6	28	28	✓
✓	4 7 5 1 2 1 3 4 1	22	22	✓

RESULT: Thus the program was executed successfully.