Name: Valluru Varshini

Class: CSE – F

Reg no: 230701369

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 a,sum=0;
    scanf("%d",&a);
    if(a>=1000)
    {
        sum+=a/1000;
        a=a%1000;
    }
    if(a>=500)
    {
        sum+=a/500;
        a=a%500;

    }
    if(a>=100)
    {
        sum+=a/100;
        a=a%100;
```

```
}
if(a>=50)
{
   sum+=a/50;
   a=a%50;
}
if(a>=20)
{
   sum+=a/20;
   a=a%20;
}
if(a>=10)
{
   sum+=a/10;
   a=a%10;
}
if(a>=5)
{
   sum+=a/5;
   a=a%5;
```

```
    if(a>=2) {
    sum+=a/2;
     a=a%2;
   }
   if(a>=1)
   {
      sum+=a/1;
      a=a%1;
}
   printf("%d",sum);
```

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

```
            k=h[j];
        }
    }
    printf("%d",k);
}
```

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>
int main()
{
    int n,temp;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(a[i]<a[j])
            {
                temp=a[i];
```

```
            a[i]=a[j];
            a[j]=temp;
        }
    }
    }
    int sum=0;
    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>
int main()
{
   int n,temp;
   scanf("%d",&n);
   int a[n];
```

```
for(int i=0;i<n;i++)
{
   scanf("%d",&a[i]);
}
for(int i=0;i<n;i++)
{
   for(int j=i+1;j<n;j++)
   {
      if(a[i]>a[j])
      {
         temp=a[i];
         a[i]=a[j];
         a[j]=temp;
      }
   }
}
   int sum=0;
   for(int i=0;i<n;i++)
   {
      sum+=(a[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	✔

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>
int main()
{
   int n,sum=0,temp=0;
   scanf("%d",&n);
```

```
int a[n];
for(int i=0;i<n;i++)
{
   scanf("%d",&a[i]);
}
int b[n];
for(int i=0;i<n;i++)
{
   scanf("%d",&b[i]);
}
for(int i=0;i<n;i++)
{
   for(int j=i+1;j<n;j++)
   {
      if(a[i]<a[j])
      {
         temp=a[i];
         a[i]=a[j];
         a[j]=temp;
      }
```

```
        }
    }
    for(int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(b[i]>b[j])
            {
                temp=b[i];
                b[i]=b[j];
                b[j]=temp;
            }
        }
    }
    for(int i=0;i<n;i++)
    {
        sum+=a[i]*b[i];
    }
    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.