Lab-3: Divide and Conquer-I

Quick Sort:

```
#include <iostream>
using namespace std;
class sorting{
   public:
   void Quick_Sort(int a[],int l,int r,int n)
   {
      if(l<r){
      int p=partition(a,l,r,n);
      cout<<endl<<"This is the Pivot Element:"<<a[p]<<endl;
      Quick_Sort(a,l,p-1,n);
      Quick_Sort(a,p+1,r,n);
      }
   }
   int partition(int a[],int l,int r,int n)
   {
      int x=a[r];
      int i=l-1;
      for(int j=l;j<=r-1;j++)
      {
         cout<<endl<<"Iteration"<<j+1<<":";
         if(a[j]<=x)
         {
            i++;
            swap(a[i],a[j]);
         }
         display(a,n);
      }
      cout<<endl<<"Swapping Element:"<<a[i+1]<<endl;
      swap(a[i+1],a[r]);
      return i+1;
   }
   void display(int a[],int n)
   {
      for(int i=0;i<n;i++)
      {
         cout<<a[i]<<" ";
      }
   }
};
int main() {
```

```
   srand(time(0));
   int n;
   cout<<"Enter Total Numbers:"<<endl;
   cin>>n;
    int arr[n];
    int choice;
    cin>>choice;
    switch(choice)
    {
      case 1: // for increasing-order
    for(int i=0;i<n;i++)
    {
        arr[i]=i+1;
    }
    break;

    case 2: // for decreasing order
    for(int i=0;i<n;i++)
    {
        arr[i]=n-i;
    }
    break;

    case 3: //for random numbers
    for(int i=0;i<n;i++)
    {
        arr[i]=rand() % 100;
    }
    break;

    default:
    return 0;
    }
   sorting s;
   cout<<"Before Sorting:"<<endl;
   s.display(arr,n);
   cout<<endl;
   s.Quick_Sort(arr,0,n-1,n);
   cout<<endl<<"After Sorting:"<<endl;
   s.display(arr,n);
    return 0;
}
```

O/P:
for increasing-order:

```
Enter Total Numbers:
5
1
Before Sorting:
1 2 3 4 5

Iteration1:1 2 3 4 5
Iteration2:1 2 3 4 5
Iteration3:1 2 3 4 5
Iteration4:1 2 3 4 5
Swapping Element:5

This is the Pivot Element:5

Iteration1:1 2 3 4 5
Iteration2:1 2 3 4 5
Iteration3:1 2 3 4 5
Swapping Element:4

This is the Pivot Element:4

Iteration1:1 2 3 4 5
Iteration2:1 2 3 4 5
Swapping Element:3

This is the Pivot Element:3

Iteration1:1 2 3 4 5
Swapping Element:2

This is the Pivot Element:2

After Sorting:
1 2 3 4 5
```

For decreasing Order:

```
Enter Total Numbers:
3
2
Before Sorting:
3 2 1

Iteration1:3 2 1
Iteration2:3 2 1
Swapping Element:3

This is the Pivot Element:1

Iteration2:1 2 3
Swapping Element:3

This is the Pivot Element:3

After Sorting:
1 2 3
```

For random order:

```
Enter Total Numbers:
3
3
Before Sorting:
74 77 51

Iteration1:74 77 51
Iteration2:74 77 51
Swapping Element:74

This is the Pivot Element:51

Iteration2:51 77 74
Swapping Element:77

This is the Pivot Element:74

After Sorting:
51 74 77
```

Merge Sort:

```
#include <iostream>
using namespace std;
class sorting{
   public:
   void Merge_Sort(int a[],int l,int r,int n)
   {
    if(l<r)
    {
        int m=l+(r-l)/2;
        cout<<endl<<"Mid Element:"<<a[m]<<endl;
        cout<<"Partition:1"<<endl;
        display(a,l,m);
        cout<<endl;
        Merge_Sort(a,l,m,n);
        cout<<"Partition:2"<<endl;
        display(a,m+1,r);
        Merge_Sort(a,m+1,r,n);
        merge(a,l,m,r,n);
    }
   }
   void merge(int a[],int l,int m,int r,int n)
   {
       int n1=m-l+1;
       int n2=r-m;
       int left[n1];
       int right[n2];
       for(int i=0;i<n1;i++)
       {
           left[i]=a[l+i];
       }
       for(int i=0;i<n2;i++)
       {
           right[i]=a[m+i+1];
       }
       int i=0,j=0;
       int k=l;
      while (i < n1 && j < n2) {
           if (left[i] <= right[j]) {
               a[k] = left[i];
               i++;
           } else {
               a[k] = right[j];
               j++;
```

```
            }
            k++;
        }
        while(i<n1)
        {
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }
    void display(int a[],int n)
    {
        for(int i=0;i<n;i++)
        {
            cout<<a[i]<<" ";
        }
    }
    void display(int a[],int l,int r)
    {
        for(int i=l;i<r;i++)
        {
            cout<<a[i]<<" ";
        }
    }
};
int main() {
    srand(time(0));
   int n;
   cout<<"Enter Total Numbers:"<<endl;
   cin>>n;
    int arr[n];
    int choice;
    cin>>choice;
    switch(choice)
    {
      case 1: // for increasing-order
    for(int i=0;i<n;i++)
    {
```

```
        arr[i]=i+1;
    }
    break;

    case 2: // for decreasing order
    for(int i=0;i<n;i++)
    {
        arr[i]=n-i;
    }
    break;

    case 3: //for random numbers
    for(int i=0;i<n;i++)
    {
        arr[i]=rand() % 100;
    }
    break;

    default:
    return 0;
    }
    sorting s;
    cout<<"Before Sorting:"<<endl;
    s.display(arr,n);
    cout<<endl;
    s.Merge_Sort(arr,0,n-1,n);
    cout<<endl<<"After Sorting:"<<endl;
    s.display(arr,n);
    return 0;
}
```

for increasing-order:

```
Enter Total Numbers:
5
1
Before Sorting:
1 2 3 4 5

Mid Element:3
Partition:1
1 2

Mid Element:2
Partition:1
1

Mid Element:1
Partition:1

4
Mid Element:4
Partition:1


After Sorting:
1 2 3 4 5
```

For Deceasing Order:

```
Enter Total Numbers:
5
2
Before Sorting:
5 4 3 2 1

Mid Element:3
Partition:1
5 4

Mid Element:4
Partition:1
5

Mid Element:5
Partition:1

2
Mid Element:2
Partition:1


After Sorting:
1 2 3 4 5
```

For Random Order:

```
Enter Total Numbers:
5
3
Before Sorting:
59 26 90 95 8

Mid Element:90
Partition:1
59 26

Mid Element:26
Partition:1
59

Mid Element:59
Partition:1

95
Mid Element:95
Partition:1


After Sorting:
8 26 59 90 95
```

Largest sum-subarray using Divide and Conquer:

```
#include <iostream>
#include<climits>
using namespace std;
class sorting{
   public:
   int Merge_Sort(int a[],int l,int r,int n)
   {
    if(l<r)
    {
       int m=l+(r-l)/2;
       int x= Merge_Sort(a,l,m,n);
       int y=Merge_Sort(a,m+1,r,n);
       int lsum=INT_MIN,rsum=INT_MIN,sum=0;
       for(int i=m;i>=l;i--)
       {
          sum+=a[i];
          lsum=max(lsum,sum);
       }
```

```
        sum=0;
        for(int i=m+1;i<=r;i++)
        {
            sum+=a[i];
            rsum=max(rsum,sum);
        }
        sum=max(x,max(y,lsum+rsum));
        return sum;
     }
     else{
         return a[l];
     }
    }
    void display(int a[],int n)
    {
        for(int i=0;i<n;i++)
        {
            cout<<a[i]<<" ";
        }
    }
};
int main() {
    srand(time(0));
   int n;
   cout<<"Enter Total Numbers:"<<endl;
   cin>>n;
    int arr[n];
    cout<<"Enter Values:"<<endl;
    for(int i=0;i<n;i++)
    {
        cin>>arr[i];
    }
    sorting s;
    cout<<"Array:"<<endl;
    s.display(arr,n);
    cout<<endl;
    int maximum=s.Merge_Sort(arr,0,n-1,n);
    cout<<"Maximum Subarray Sum:"<<maximum;
    return 0;
}
```

O/P:

```
Enter Total Numbers:
5
Enter Values:
5
4
1
7
8
Array:
5 4 1 7 8
Maximum Subarray Sum:25
```