EXPERIMENT NO. 10

Ques 1 :- Write a program to implement Insertion Sort of an array of 'n' elements using a random function. Share the time complexity and n should be large enough to see the difference in execution.

```
#include<stdio.h>
#include<stdlib.h>
void insertionSort(int arr[],int size)
{
   int i,j,x;
   for(i=1;i<size;i++)
   {
      j=i-1;
      x=arr[i];
      while(j>-1 && arr[j]>x)
      {
         arr[j+1]=arr[j];
         j--;
      }
      arr[j+1]=x;
   }
}
int main()
```

```
{
   int size;
   printf("Enter the size of the array :\n");
   scanf("%d",&size);
   int arr[size];
   printf("Enter the elements in array :\n");
   for(int i=0;i<size;i++)
   {
      scanf("%d",&arr[i]);
   }
   insertionSort(arr,size);
   printf("Sorted Array is :\n");
   for(int i=0;i<size;i++)
   {
      printf("%d ",arr[i]);
   }
   return 0;
}
```

```
Enter the size of the array :
6
Enter the elements in array :
89
54
34
22
11
9
Sorted Array is :
9 11 22 34 54 89
PS E:\Data Structure and Algorithm In C\Sorting>
```

Time Complexity of Insertion Sort is O(n^2).

Ques 2 :- Write a program to implement Selection Sort of an array of 'n' elements using a random function. Share the time complexity and n should be large enough to see the difference in execution.

```
#include<stdio.h>
#include<stdlib.h>
void swap(int *x, int *y)
{
   int temp = *x;
   *x = *y;
   *y = temp;
}
```

```
void selectionSort(int arr[],int size)
{
   int i,j,k;
   for(i=0;i<size-1;i++)
   {
      for(j=k=i;j<size;j++)
      {
         if(arr[j]<arr[k])
         {
            k=j;
         }
      }
      swap(&arr[i],&arr[k]);
   }
}
int main()
{
   int size;
   printf("Enter the size of the array :\n");
   scanf("%d",&size);
   int arr[size];
   printf("Enter the elements in array :\n");
```

```
    for(int i=0;i<size;i++)
    {
        scanf("%d",&arr[i]);
    }
    selectionSort(arr,size);
    printf("Sorted Array is :\n");
    for(int i =0;i<size;i++)
    {
        printf("%d ",arr[i]);
    }
}
```

```
Enter the size of the array :
6
Enter the elements in array :
89
56
78
43
23
66
Sorted Array is :
23 43 56 66 78 89
PS E:\Data Structure and Algorithm In C\Sorting>
```

Time Complexity of Selection Sort is O(n^2).

Ques 3 :- Write a program to implement Merge Sort of an array of 'n' elements using a random function. Share the time complexity and n should be large enough to see the difference in execution.

```
#include<stdio.h>

#include<stdlib.h>

void merge(int arr[], int s, int e) {

        int mid = (s+e)/2;

        int lenLeft = mid - s + 1;
        int lenRight = e - mid;

        int *left = (int *)malloc(lenLeft * sizeof(int));
    int *right = (int *)malloc(lenRight * sizeof(int));

        int k = s;
        for(int i=0; i<lenLeft; i++) {
                left[i] = arr[k];
                k++;
        }

        k = mid+1;
```

```
for(int i=0; i<lenRight; i++) {
        right[i] = arr[k];
        k++;
}



int leftIndex = 0;
int rightIndex = 0;
int mainArrayIndex = s;

while(leftIndex < lenLeft &&  rightIndex < lenRight) {

        if(left[leftIndex] < right[rightIndex] ) {
                arr[mainArrayIndex] =  left[leftIndex];
                mainArrayIndex++;
                leftIndex++;
        }
        else {
                arr[mainArrayIndex] =  right[rightIndex];
                mainArrayIndex++;
                rightIndex++;
        }
```

```
        }



        while(rightIndex < lenRight) {
                arr[mainArrayIndex] =  right[rightIndex];
                mainArrayIndex++;
                rightIndex++;
        }



        while(leftIndex < lenLeft) {
                arr[mainArrayIndex] =  left[leftIndex];
                mainArrayIndex++;
                leftIndex++;
        }
        free(left);
        free(right);
}

void mergeSort(int arr[], int s, int e) {
        if(s >= e) {
```

```
            return;
        }
        int mid = (s+e)/2;
        //recusive call for left array
        mergeSort(arr,s, mid);
        //recusive call for right array
        mergeSort(arr, mid+1, e);
        //merge 2 sorted arrays
        merge(arr, s, e);
}

int main() {

        int size;
    printf("Enter the size of the array :\n");
    scanf("%d",&size);
    int arr[size];
    printf("Enter the elements in array :\n");
    for(int i=0;i<size;i++)
    {
        scanf("%d",&arr[i]);
    }
```

```
        int s = 0;
        int e = size - 1;
    printf("Before Merge Sort :\n");
    for(int i=0;i<size;i++)
    {
        printf("%d ",arr[i]);
    }
    printf("\n");
    mergeSort(arr,s,e);
    printf("After Merge Sort :\n");
    for(int i=0;i<size;i++)
    {
        printf("%d ",arr[i]);
    }
        return 0;
}
```

```
Enter the size of the array :
5
Enter the elements in array :
90
80
66
5
4
Before Merge Sort :
90 80 66 5 4
After Merge Sort :
4 5 66 80 90
```

Time Complexity of Merge Sort is O(nlogn).

Ques 4 :- Write a program to implement Quick Sort of an array of 'n' elements using a random function. Share the time complexity and n should be large enough to see the difference in execution.

```
#include<stdio.h>
#include<stdlib.h>
void swap(int*x,int*y)
{
   int temp = *x;
   *x=*y;
   *y=temp;
}
```

```
int partition(int arr[],int l,int h)
{
   int pivot = arr[l];
   int i=l,j=h;
   do
   {
      do
      {
         i++;
      } while (arr[i]<=pivot);
      do
      {
         j--;
      } while (arr[j]>pivot);

      if(i<j)
      {
         swap(&arr[i],&arr[j]);
      }
   } while (i<j);
   swap(&arr[l],&arr[j]);
   return j;
```

```
}
void quickSort(int arr[],int l, int h)
{
   int j;
   if(l<h)
   {
      j = partition(arr,l,h);
      quickSort(arr,l,j);
      quickSort(arr,j+1,h);
   }

}
int main()
{
   int size;
   printf("Enter the size of the array :\n");
   scanf("%d",&size);
   int arr[size];
   printf("Enter the elements in array :\n");
   for(int i=0;i<size;i++)
   {
      scanf("%d",&arr[i]);
```

```
    }
    int l=0,h=size;
    quickSort(arr,l,h);
    printf("Sorted Array is :\n");
    for(int i=0;i<size;i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

```
Enter the size of the array :
5
Enter the elements in array :
10
70
60
56
44
Sorted Array is :
10 44 56 60 70
```

Time Complexity of Quick sort is O(nlogn).