ADA Lab Programs

1. Selection sort

```
// C program for implementation of selection sort
#include <stdio.h>
#include<stdlib.h>

#include<time.h>


void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
            min_idx = j;

        // Swap the found minimum element with the first element
           if(min_idx != i)
            swap(&arr[min_idx], &arr[i]);
    }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver program to test above functions
int main()
{
```

```
    int n, i;
clock_t start, end;
double time_taken;
printf("Enter the number of elements to be sorted: ");
scanf("%d", &n);
int arr[n];
printf("Generating %d random elements...\n", n);
srand(time(NULL)); // Seed for random number generator
for(i=0;i<n;i++)
{
     arr[i] = rand() % 10000; // Generate random integers between
                                    0 and 9999
}
printf("Sorting the array using Selection Sort...\n");
start = clock(); // Start timer
selectionsort(arr,n);
end = clock(); // Stop timer
printf("Sorted array: \n”);
for(i=0;i<n;i++) {
printf("%d ", arr[i]); }
time_taken = ((double) (end - start)) / CLOCKS_PER_SEC;
printf("\nTime taken to sort %d elements: %lf seconds", n,
time_taken);


return 0;
}
```

---

2. QUICK SORT

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

#include<time.h>


void quicksort(int*, int, int);
int partition (int*, int, int);
int main()
{
    int n, i;
    clock_t start, end;
    double time_taken;
    printf("Enter the number of elements to be sorted: ");
    scanf("%d", &n);
```

```
    int arr[n];
    printf("Generating %d random elements...\n", n);
    srand(time(NULL)); // Seed for random number generator
    for(i=0;i<n;i++)
    {
        arr[i] = rand() % 10000; // Generate random integers
                                    between 0 and 9999
    }
    printf("Sorting the array using Quick Sort...\n");
    start = clock(); // Start timer
    quicksort(arr, 0, n - 1);
    end = clock(); // Stop timer
    printf("Sorted array: \n”);
    for(i=0;i<n;i++)
    {
        printf("%d ", arr[i]);
    }
    time_taken = ((double) (end - start)) / CLOCKS_PER_SEC;
    printf("\nTime taken to sort %d elements: %lf seconds", n,
    time_taken);

    return 0;
}

void quicksort(int arr[],int low,int high)
{

    if (low < high)
   {
       int j = partition(arr, low, high);

       quicksort(arr, low, j-1);
       quicksort(arr, j+1, high);
   }
}

int partition (int arr[],int low,int high)
{
    int pivot = arr[low];
    int temp;
    int i = low;
    int j = high+1;
    while(i < j)
    {
        do
        {
            i++;
        }while (arr[i] < pivot && i<high);
        do
```

```
        {
            j--;
        } while (arr[j] >= pivot && j>low);
        if(i<j)
        {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    temp = arr[low];
    arr[low] = arr[j];
    arr[j] = temp;

    return j;
}
```

3. MERGE SORT

```
#include<stdio.h>

#include<stdlib.h>

#include<time.h>

void merge_sort(int[], int, int);

void merge(int[], int, int, int);

int main()

{

    int n, i;
    clock_t start, end;

    double time_taken;

    printf("Enter the number of elements to be sorted:
            ");

    scanf("%d", &n);

    int arr[n];
    printf("Generating %d random elements...\n", n);
    srand(time(NULL)); // Seed for random number
                              generator
```

```
    for(i=0;i<n;i++)

    {

        arr[i] = rand() % 10000; // Generate random
                                 integers between 0 and 9999

    }

    printf("Sorting the array using Merge Sort...\n");

    start = clock(); // Start timer
    merge_sort(arr, 0, n - 1);
    end = clock(); // Stop timer

    printf("Sorted array: \n");

    for(i=0;i<n;i++)

    {
        printf("%d ", arr[i]);

    }

    time_taken = ((double) (end - start)) /
CLOCKS_PER_SEC;

    printf("\nTime taken to sort %d elements: %lf
            seconds", n, time_taken);

    return 0;

}

void merge_sort(int arr[], int low, int high)

{

    int mid;

    if(low < high)

    {

        mid=(low+high)/2;

        merge_sort(arr, low, mid);

        merge_sort(arr, mid + 1, high);
```

```
        merge(arr, low, mid, high);
    }
}
void merge(int arr[], int low, int mid, int high)
{
    inti=low,j=mid+1,k=0,temp[high-low+1];
    while(i <= mid && j <= high)
    {
        if(arr[i] <= arr[j])
        {
            temp[k++] = arr[i++];
        }
        else
        {
            temp[k++] = arr[j++];
        }
    }
    while(i <= mid)
    {
        temp[k++] = arr[i++];
    }
    while(j <= high)
    {
        temp[k++] = arr[j++];
    }
```

```
    for(i=low,k=0;i<=high;i++,k++)
    {
        arr[i] = temp[k];
    }
}
```