Name:- Suryakant Upadhyay

Prn :- 20220802043

Roll No :- A20

DAA Assignment -2

Topic - Divide and conquer algorithms

1. Write a C code to implement Binary Search.

```
#include <stdio.h>

int binarySearch(int arr[], int size, int key) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            left = mid + 1;
        else
            right = mid - 1;
    }

    return -1; // Key not found
}

int main() {
    int arr[] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key = 12;
    int result = binarySearch(arr, size, key);

    if (result != -1)
        printf("Element found at index: %d\n", result);
    else
        printf("Element not found\n");

    return 0;
}
```

2. Write a C code to implement merge Sort.

```
#include <stdio.h>

void merge(int arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    i = 0;
    j = 0;
    k = left;

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
```

```
        merge(arr, left, mid, right);
    }
}

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: \n");
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);

    mergeSort(arr, 0, size - 1);

    printf("\nSorted array: \n");
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

3. Write a C code implement Quick sort.

```
#include <stdio.h>

void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
```

```
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: \n");
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);

    quickSort(arr, 0, size - 1);

    printf("\nSorted array: \n");
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);

    return 0;
}
```