Name- Chiradeep Banik.

Enrol. No. - 20UCS176

Branch - Computer Sc. and Engineering

Section - A

Assignment - 2

Merge sort

```
#include <stdio.h>

void merge(int* arr, int start, int mid, int end){
    int len1 = mid-start+1;
    int len2 = end - mid;

    int leftArr[len1], rightArr[len2];

    for(int i=0; i<len1; i++){
        leftArr[i] = arr[start+i];
    }
    for(int j=0; j<len2; j++){
        rightArr[j] = arr[mid+1+j];
    }
}
```

→

```
int i=0, j=0, k=start;
while (i<len1 && j<len2) {
    if (leftArr[i] <= rightArr[j]) {
        arr[k] = leftArr[i];
        i++;
    } else {
        arr[k] = rightArr[j];
        j++;
    }
    k++;
}
while (i<len1) {
    arr[k] = leftArr[i];
    i++;
    k++;
}
while (j<len2) {
    arr[k] = rightArr[j];
    j++;
    k++;
}
void mergeSort (int* arr, int start, int end) {
    if (start<end) {
        int mid = start + (end-start)/2;
        mergeSort (arr, start, mid);
        mergeSort (arr, mid+1, end);
        merge (arr, start, mid, end);
    }
}
```

```
void main(){
    int arr[]={4,2,5,3,1};
    int size = sizeof(arr) / sizeof(arr[0]);
    mergeSort(arr, 0, size-1);
    for(int i=0; i<size; i++){
        printf("%d", arr[i]);
    }
}
```

Output

→ 1 2 3 4 5

Quick Sort

```
#include <stdio.h>

void Quick Sort (int* arr, int first, int last){

    int i, j, pivot, temp;
    if (first < last){
        pivot = first;
        i = first;
        j = last;

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

→

```
    quickSort(arr, first, j-1);
    quickSort(arr, j+1, last);
  }
}

int main(){
  int arr[] = {3, 1, 4, 5, 2};
  int size = sizeof(arr)/sizeof(arr[0]);
  quickSort(arr, 0, size-1);
  for(int i=0; i<size; i++){
    printf("%d", arr[i]);
  }
}
```

Output

↳ 1 2 3 4 5

Time Complexity Comparison

	Best Case	Avg. Case	Worst Case
Merge Sort	$O(n * \log n)$	$O(n * \log n)$	$O(n * \log n)$
Quick Sort.	$O(n * \log n)$	$O(n * \log n)$	$O(n^2)$

Bindeep Banik