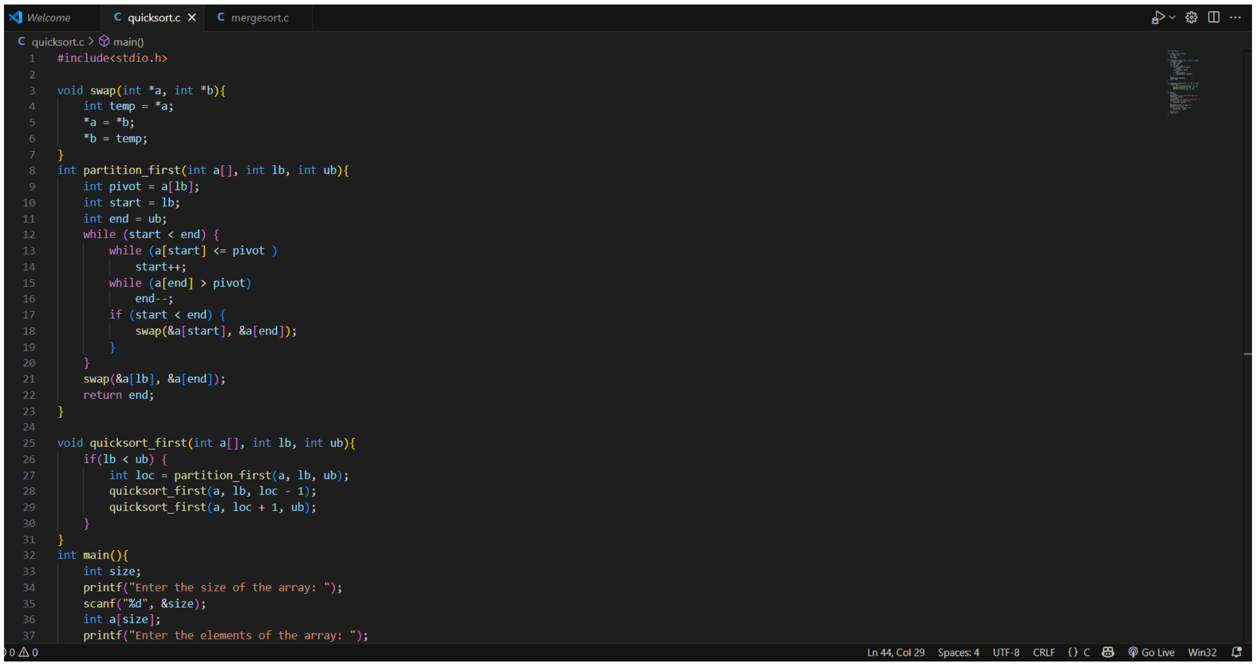
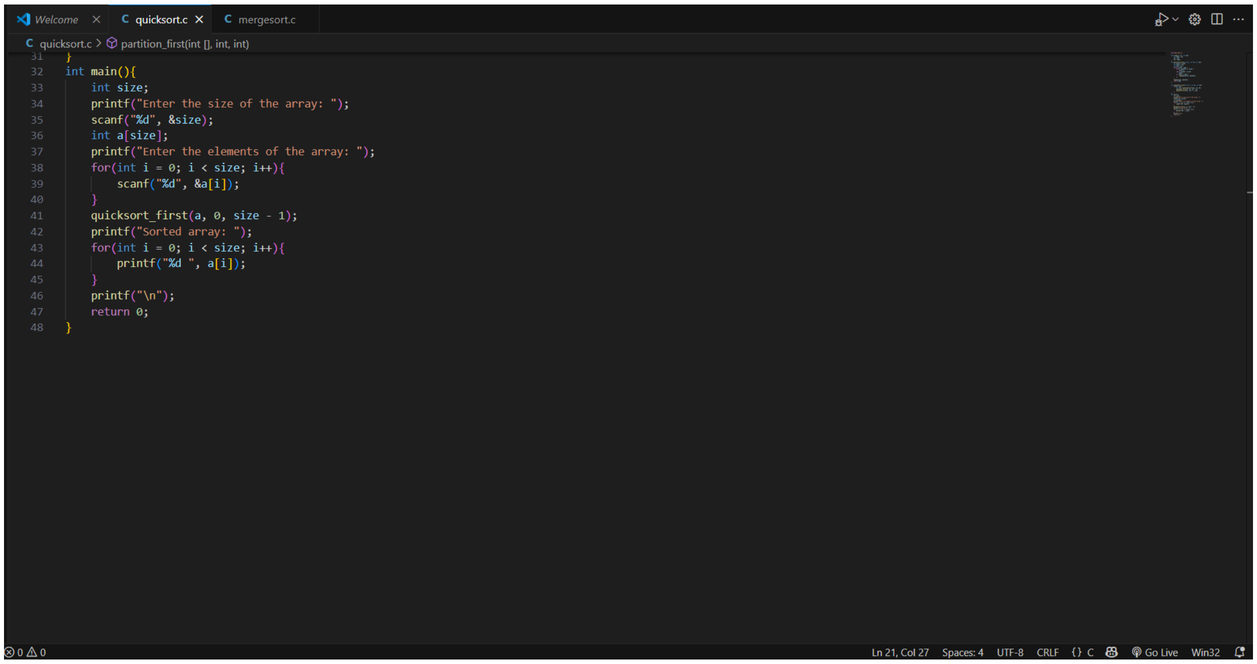
Name: Mithlesh Yeole
Roll no.: B3-B3-59
PRACTICAL 8

QUICK SORT implementation:
Code:

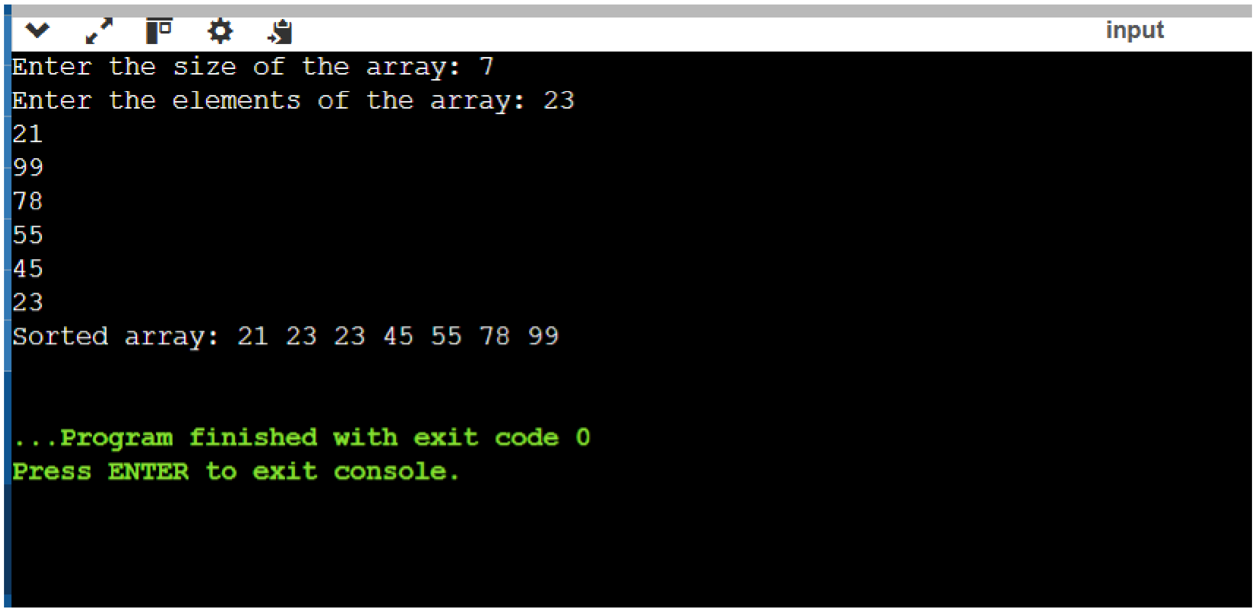

Welcome
quicksort.c
mergesort.c
quicksort.c > main()
Ln 44, Col 29 Spaces: 4 UTF-8 CRLF { } C Go Live Win32

```c
#include<stdio.h>

void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partition_first(int a[], int lb, int ub){
    int pivot = a[lb];
    int start = lb;
    int end = ub;
    while (start < end) {
        while (a[start] <= pivot )
            start++;
        while (a[end] > pivot)
            end--;
        if (start < end) {
            swap(&a[start], &a[end]);
        }
    }
    swap(&a[lb], &a[end]);
    return end;
}

void quicksort_first(int a[], int lb, int ub){
    if(lb < ub) {
        int loc = partition_first(a, lb, ub);
        quicksort_first(a, lb, loc - 1);
        quicksort_first(a, loc + 1, ub);
    }
}
int main(){
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int a[size];
    printf("Enter the elements of the array: ");
```


Welcome
quicksort.c
mergesort.c
quicksort.c > partition_first(int [], int, int)
Ln 21, Col 27 Spaces: 4 UTF-8 CRLF { } C Go Live Win32

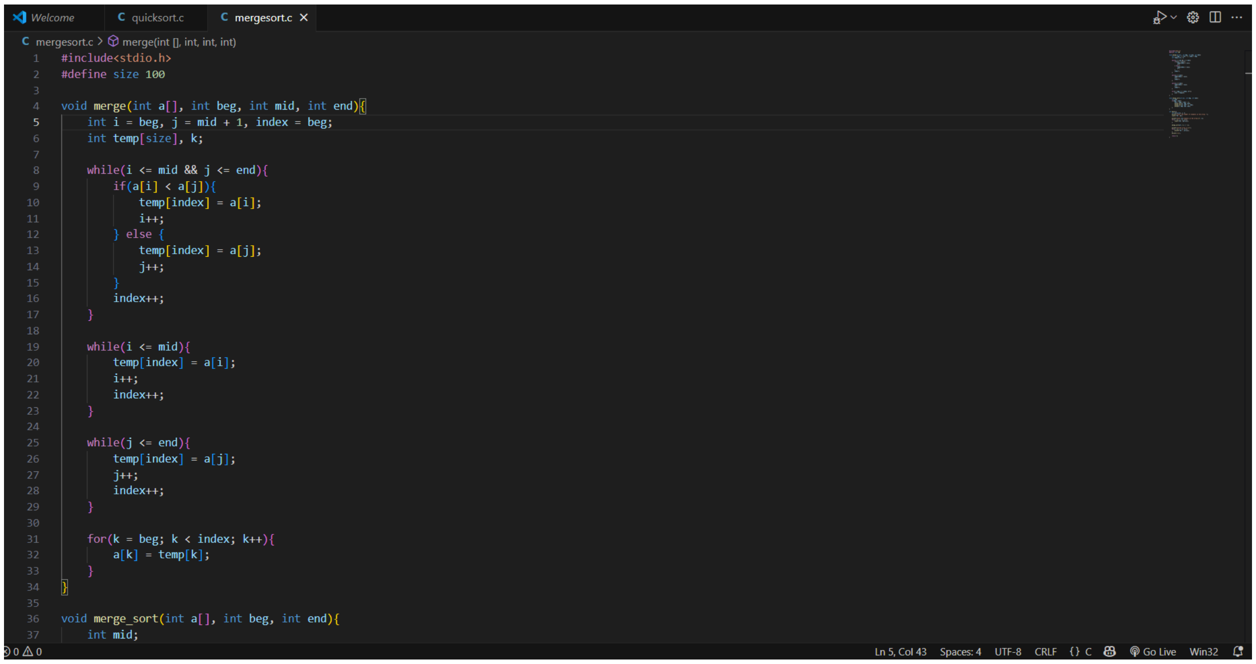
```c
}
int main(){
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int a[size];
    printf("Enter the elements of the array: ");
    for(int i = 0; i < size; i++){
        scanf("%d", &a[i]);
    }
    quicksort_first(a, 0, size - 1);
    printf("Sorted array: ");
    for(int i = 0; i < size; i++){
        printf("%d ", a[i]);
    }
    printf("\n");
    return 0;
}
```

Output:

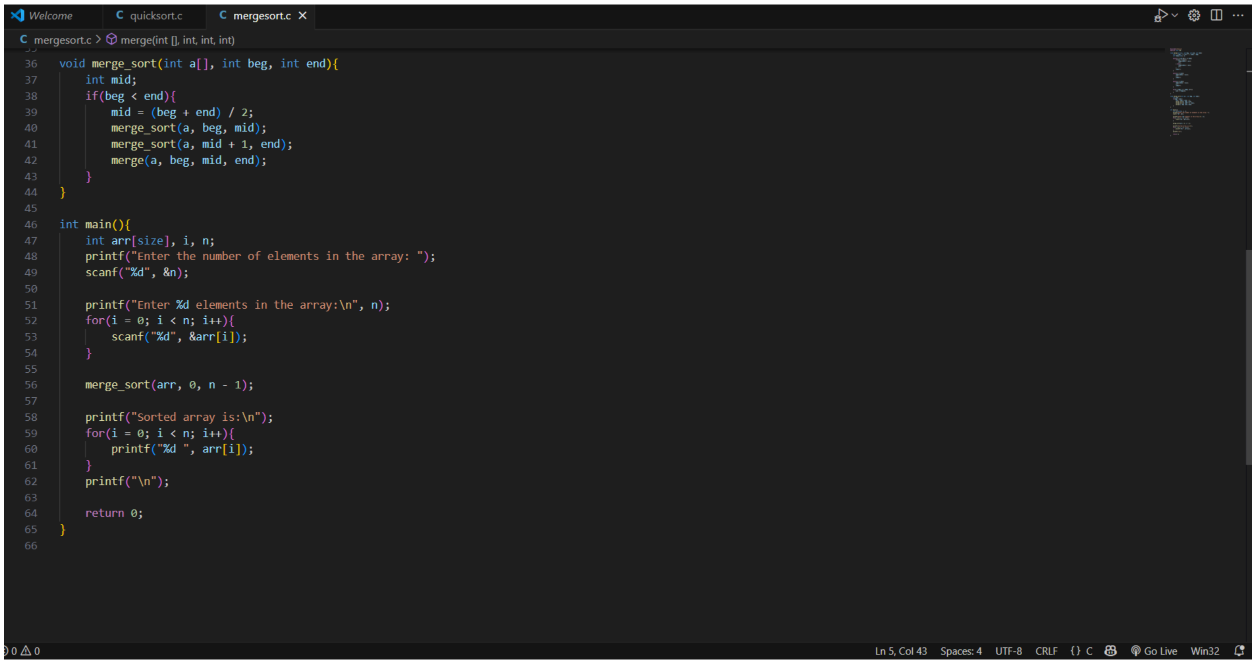


```
Enter the size of the array: 7
Enter the elements of the array: 23
21
99
78
55
45
23
Sorted array: 21 23 23 45 55 78 99


...Program finished with exit code 0
Press ENTER to exit console.
```

MERGE SORT implementation:

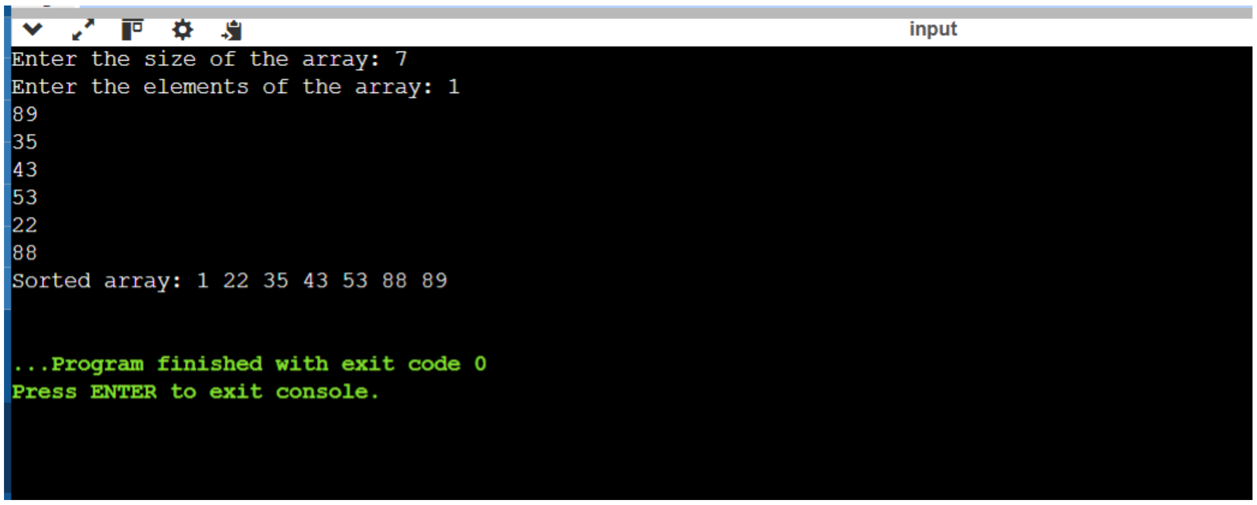

Welcome
quicksort.c
mergesort.c
mergesort.c > merge(int [], int, int, int)
Ln 5, Col 43 Spaces: 4 UTF-8 CRLF {} C Go Live Win32

```c
#include<stdio.h>
#define size 100

void merge(int a[], int beg, int mid, int end){
    int i = beg, j = mid + 1, index = beg;
    int temp[size], k;

    while(i <= mid && j <= end){
        if(a[i] < a[j]){
            temp[index] = a[i];
            i++;
        } else {
            temp[index] = a[j];
            j++;
        }
        index++;
    }

    while(i <= mid){
        temp[index] = a[i];
        i++;
        index++;
    }

    while(j <= end){
        temp[index] = a[j];
        j++;
        index++;
    }

    for(k = beg; k < index; k++){
        a[k] = temp[k];
    }
}

void merge_sort(int a[], int beg, int end){
    int mid;
```


Welcome
quicksort.c
mergesort.c
mergesort.c > merge(int [], int, int, int)
Ln 5, Col 43 Spaces: 4 UTF-8 CRLF {} C Go Live Win32

```c
void merge_sort(int a[], int beg, int end){
    int mid;
    if(beg < end){
        mid = (beg + end) / 2;
        merge_sort(a, beg, mid);
        merge_sort(a, mid + 1, end);
        merge(a, beg, mid, end);
    }
}

int main(){
    int arr[size], i, n;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

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

    merge_sort(arr, 0, n - 1);

    printf("Sorted array is:\n");
    for(i = 0; i < n; i++){
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Output:



```
Enter the size of the array: 7
Enter the elements of the array: 1
89
35
43
53
22
88
Sorted array: 1 22 35 43 53 88 89


...Program finished with exit code 0
Press ENTER to exit console.
```