merge_sort.c | quick_sort.c ●

quick_sort.c

```c
/*Sort a given set of N integer elements using the Quick sort technique and also compute time taken for different N values.*/
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<time.h>
clock_t start,end;
double cpu_time;

void quick_sort(int [] ,int,int);
int partition(int [] ,int,int);

int main()
{
    int a[10000],n,i,j;
    srand(time(0));
    printf("Enter number of elements:\n");
    scanf("%d",&n);
    printf("Array elements:\n");
    for(i=0;i<n;i++)
    {
       a[i]=rand()%100;
       printf("%d ",a[i]);
    }
    start= clock();
    quick_sort(a,0,n-1);
    printf("\nSorted Array:\n");
    for(i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
    end = clock();
    cpu_time = (double)(end - start) / CLOCKS_PER_SEC;
    printf("\nExecution time for Quick Sort = %f ms\n", cpu_time*1000);
    getch();
}
void quick_sort(int a[],int low,int high)
{
    int mid;
```

merge_sort.c | quick_sort.c ●

quick_sort.c

```c
{
    int mid;
    if(low<high)
    {
        mid=partition(a,low,high);
        quick_sort(a,low,mid-1);
        quick_sort(a,mid+1,high);
    }
}

int partition(int a[],int low,int high)
{
    int i,j,temp,pivot;
    pivot=a[low];
    i=low+1;
    j=high;
    while(i<=j)
    {
        while(a[i]<=pivot)
        {
            i++;
        }
        while(a[j]>pivot)
        {
            j--;
        }
        if(i<j)
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }
    temp=a[low];
    a[low]=a[j];
    a[j]=temp;
    return j;
}
```

merge_sort.c | quick_sort.c ●

quick_sort.c

```
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }
    temp=a[low];
    a[low]=a[j];
    a[j]=temp;
    return j;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: a

```
PS C:\Users\muska\OneDrive\Desktop\C programs> gcc quick_sort.c
PS C:\Users\muska\OneDrive\Desktop\C programs> .\a.exe
Enter number of elements:
60
Array elements:
2 2 96 19 6 76 99 59 18 54 32 48 96 94 22 51 45 16 69 73 9 78 9 77 39 47 56 50 85 75 63 98 34 94 68 24 2 30 70 68 26 50 9 36 57 85 20 17 77 40 13 24 50 47 99 27 20 73 97 14
Sorted Elements
2 2 2 6 9 9 9 13 14 16 17 18 19 20 20 22 24 24 26 27 30 32 34 36 39 40 45 47 47 48 50 50 50 51 54 56 57 59 63 68 68 69 70 73 73 75 76 77 77 78 85 85 94 94 96 96 97 98 99 99
Execution time for Quick Sort = 5.000000 ms
PS C:\Users\muska\OneDrive\Desktop\C programs> gcc quick_sort.c
PS C:\Users\muska\OneDrive\Desktop\C programs> .\a.exe
Enter number of elements:
100
Array elements:
60 66 71 7 60 21 55 37 14 61 70 40 80 16 65 57 17 90 54 33 80 22 32 68 48 97 94 1 51 58 44 43 16 86 29 84 38 48 56 15 66 59 70 72 75 34 83 1 68 85 29 13 2 27 85 39 60 87 8 40 79 38 17 57 53 37 63 69 5
67 5 54 64 10 75 49 18 83 20 55 52 58 47 53 4 39 19 35 38 98 94 81 34 45 71 58 74 7 34 54
Sorted Elements
1 1 2 4 5 5 7 7 8 10 13 14 15 16 16 17 17 18 19 20 21 22 27 29 29 32 33 34 34 34 35 37 37 38 38 38 39 39 40 40 43 44 45 47 48 48 49 51 52 53 53 54 54 54 55 55 56 57 57 58 58 58 59 60 60 60 61 63 64 65
66 66 67 68 68 69 70 70 71 71 72 74 75 75 79 80 80 81 83 83 84 85 85 86 87 90 94 94 97 98
Execution time for Quick Sort = 12.000000 ms
```

N	EXECUTION TIME(MS)
500	0.301
1000	0.58
1500	0.707
2000	0.881
2500	1.501
3000	1.298