

# VARUN BHARGAVA – 241010282

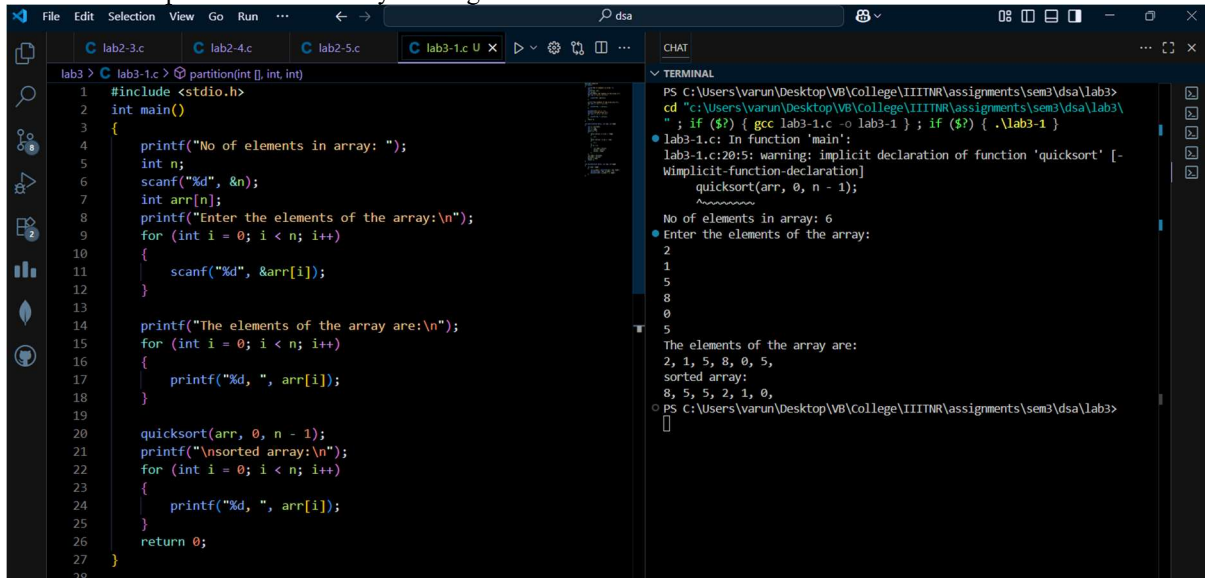
## DATA STRUCTURES TASK-3

### Task 01: Quick Sort:

( <https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab3/lab3-1.c> )

Write a program to perform the following operations using the Quick Sort algorithm:

1. Take user input to create an array of integers.



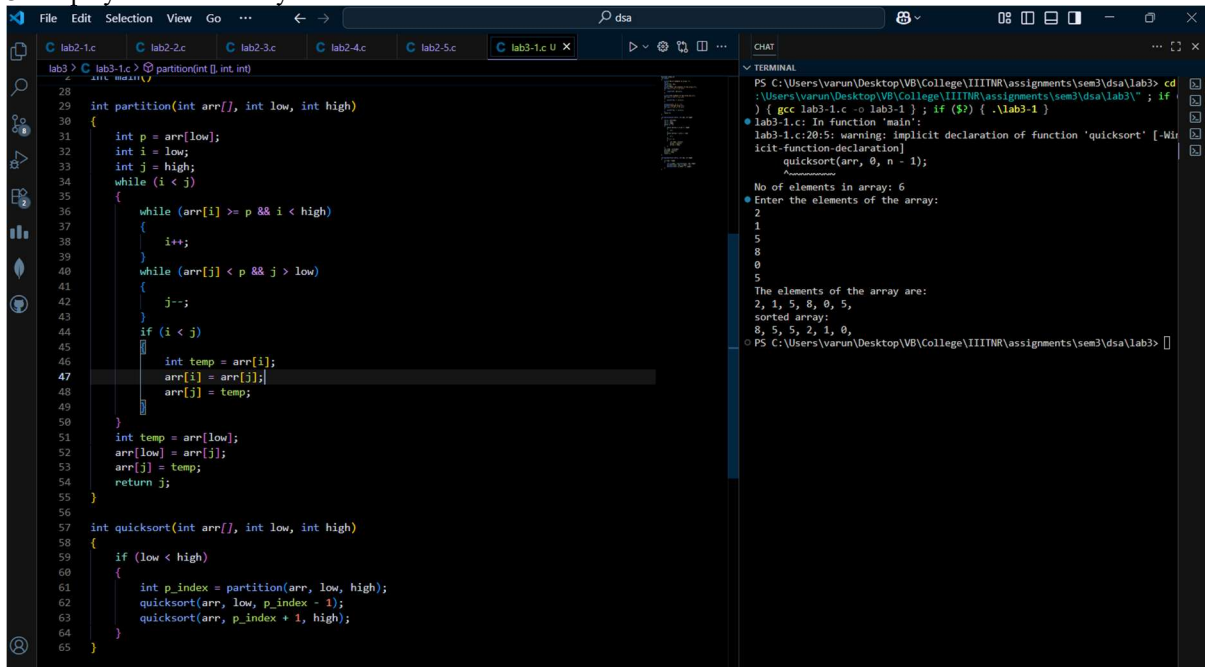
```
1 #include <stdio.h>
2 int main()
3 {
4     printf("No of elements in array: ");
5     int n;
6     scanf("%d", &n);
7     int arr[n];
8     printf("Enter the elements of the array:\n");
9     for (int i = 0; i < n; i++)
10     {
11         scanf("%d", &arr[i]);
12     }
13
14     printf("The elements of the array are:\n");
15     for (int i = 0; i < n; i++)
16     {
17         printf("%d ", arr[i]);
18     }
19
20     quicksort(arr, 0, n - 1);
21     printf("\nsorted array:\n");
22     for (int i = 0; i < n; i++)
23     {
24         printf("%d ", arr[i]);
25     }
26     return 0;
27 }
```

Terminal Output:

```
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3\"
; if ($?) { gcc lab3-1.c -o lab3-1 } ; if ($?) { .\lab3-1 }
lab3-1.c: In function 'main':
lab3-1.c:20:5: warning: implicit declaration of function 'quicksort' [-Wimplicit-function-declaration]
    quicksort(arr, 0, n - 1);
    ~~~~~
No of elements in array: 6
Enter the elements of the array:
2
1
5
8
0
5
The elements of the array are:
2, 1, 5, 8, 0, 5,
sorted array:
8, 5, 5, 2, 1, 0,
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3>
```

2. Sort the array in descending order using Quick Sort.

3. Display the sorted array.



```
28
29 int partition(int arr[], int low, int high)
30 {
31     int p = arr[low];
32     int i = low;
33     int j = high;
34     while (i < j)
35     {
36         while (arr[i] >= p && i < high)
37             i++;
38         while (arr[j] < p && j > low)
39             j--;
40         if (i < j)
41         {
42             int temp = arr[i];
43             arr[i] = arr[j];
44             arr[j] = temp;
45         }
46     }
47     int temp = arr[low];
48     arr[low] = arr[j];
49     arr[j] = temp;
50     return j;
51 }
52
53 int quicksort(int arr[], int low, int high)
54 {
55     if (low < high)
56     {
57         int p_index = partition(arr, low, high);
58         quicksort(arr, low, p_index - 1);
59         quicksort(arr, p_index + 1, high);
60     }
61 }
```

Terminal Output:

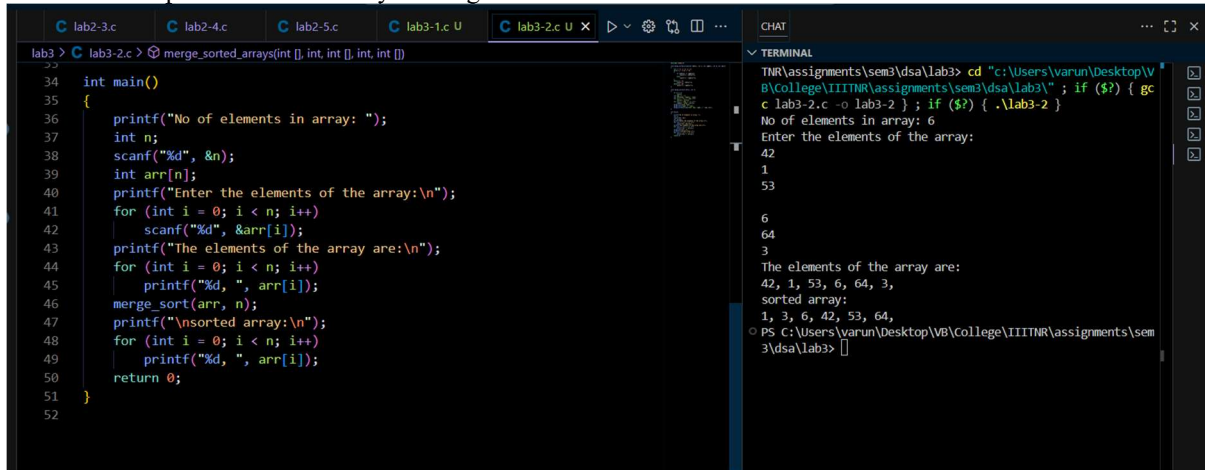
```
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3\"
; if ($?) { gcc lab3-1.c -o lab3-1 } ; if ($?) { .\lab3-1 }
lab3-1.c: In function 'main':
lab3-1.c:20:5: warning: implicit declaration of function 'quicksort' [-Wimplicit-function-declaration]
    quicksort(arr, 0, n - 1);
    ~~~~~
No of elements in array: 6
Enter the elements of the array:
2
1
5
8
0
5
The elements of the array are:
2, 1, 5, 8, 0, 5,
sorted array:
8, 5, 5, 2, 1, 0,
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3>
```

## Task 02: Merge Sort:

( <https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab3/lab3-2.c> )

Write a program to perform the following operations using Merge Sort:

1. Take user input to create an array of integers.



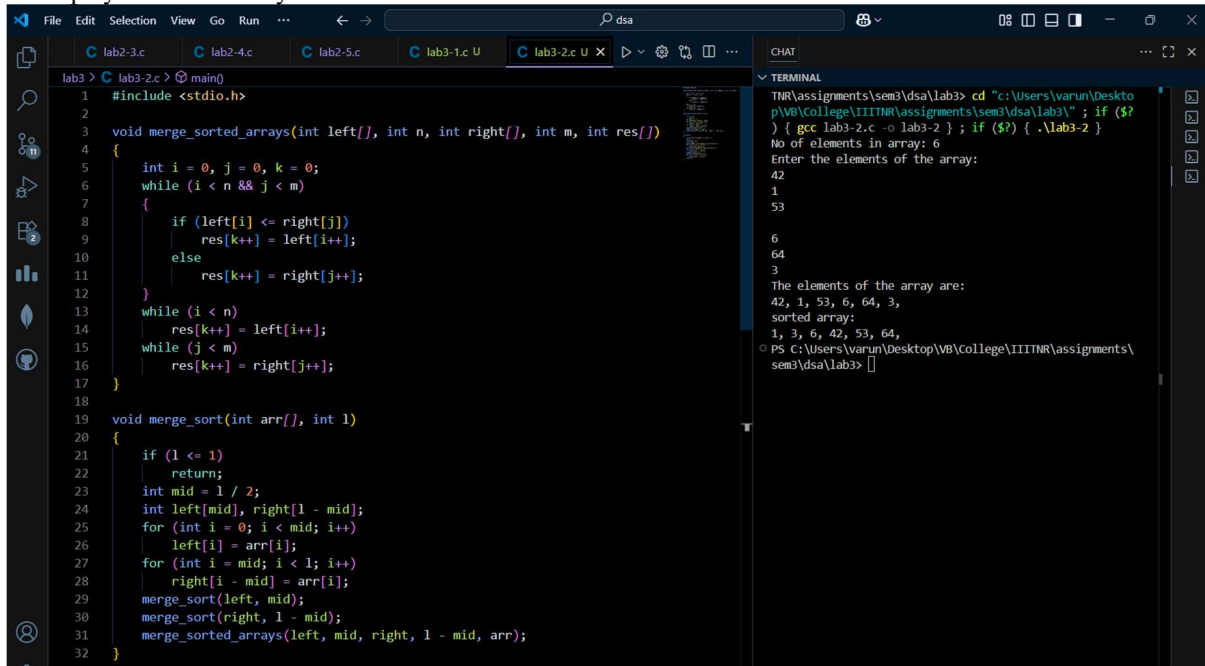
```
lab3 > C lab3-2.c > merge_sorted_arrays(int [], int [], int [], int [])
34 int main()
35 {
36     printf("No of elements in array: ");
37     int n;
38     scanf("%d", &n);
39     int arr[n];
40     printf("Enter the elements of the array:\n");
41     for (int i = 0; i < n; i++)
42     {
43         scanf("%d", &arr[i]);
44     }
45     printf("The elements of the array are:\n");
46     for (int i = 0; i < n; i++)
47     {
48         printf("%d, ", arr[i]);
49     }
50     merge_sort(arr, n);
51     printf("Unsorted array:\n");
52     for (int i = 0; i < n; i++)
53     {
54         printf("%d, ", arr[i]);
55     }
56     return 0;
57 }
```

TERMINAL

```
TNR\assignments\sem3\dsa\lab3> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3"; if ($?) { gcc lab3-2.c -o lab3-2 } ; if ($?) { .\lab3-2 }
No of elements in array: 6
Enter the elements of the array:
42
1
53
6
64
3
The elements of the array are:
42, 1, 53, 6, 64, 3,
sorted array:
1, 3, 6, 42, 53, 64,
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3>
```

2. Sort the array in ascending order using the Merge Sort algorithm.

3. Display the sorted array.



```
lab3 > C lab3-2.c > main()
1 #include <stdio.h>
2
3 void merge_sorted_arrays(int left[], int n, int right[], int m, int res[])
4 {
5     int i = 0, j = 0, k = 0;
6     while (i < n && j < m)
7     {
8         if (left[i] <= right[j])
9             res[k++] = left[i++];
10        else
11            res[k++] = right[j++];
12    }
13    while (i < n)
14        res[k++] = left[i++];
15    while (j < m)
16        res[k++] = right[j++];
17 }
18
19 void merge_sort(int arr[], int l)
20 {
21     if (l <= 1)
22         return;
23     int mid = l / 2;
24     int left[mid], right[l - mid];
25     for (int i = 0; i < mid; i++)
26         left[i] = arr[i];
27     for (int i = mid; i < l; i++)
28         right[i - mid] = arr[i];
29     merge_sort(left, mid);
30     merge_sort(right, l - mid);
31     merge_sorted_arrays(left, mid, right, l - mid, arr);
32 }
```

TERMINAL

```
TNR\assignments\sem3\dsa\lab3> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3"; if ($?) { gcc lab3-2.c -o lab3-2 } ; if ($?) { .\lab3-2 }
No of elements in array: 6
Enter the elements of the array:
42
1
53
6
64
3
The elements of the array are:
42, 1, 53, 6, 64, 3,
sorted array:
1, 3, 6, 42, 53, 64,
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3>
```

### Task 03: Hybrid Sorting Algorithm:

( <https://github.com/varunnnb/dsa-sem3-iiitr/blob/main/lab3/lab3-3.c> )

Write a program to perform the following operations using a Hybrid Sorting Algorithm:

1. Take user input to create an array of integers.

The screenshot shows a Windows IDE with a C++ program for quicksort and insertion sort. The code is in a file named 'lab3-3.c'. The program prompts the user for the number of elements in the array (10) and then for the elements themselves (32, 32, 5, 7, 9765, 53, 75, 74, 2, 7). It then displays the sorted array: 2, 5, 7, 7, 32, 32, 53, 74, 75, 9765. The terminal output shows the execution of the program.

```

lab3 > C lab3-3.c > insertion_sort(int [], int)
55 int quicksort(int arr[], int low, int high)
56 {
57     return 0;
58 }
59
60 int main()
61 {
62     printf("No of elements in array: ");
63     int n;
64     scanf("%d", &n);
65     int arr[n];
66     printf("Enter the elements of the array:\n");
67     for (int i = 0; i < n; i++)
68     {
69         scanf("%d", &arr[i]);
70     }
71     printf("The elements of the array are:\n");
72     for (int i = 0; i < n; i++)
73     {
74         printf("%d, ", arr[i]);
75     }
76     if (n <= 5)
77     {
78         insertion_sort(arr, n);
79     }
80     else
81     {
82         quicksort(arr, 0, n - 1);
83     }
84     printf("\nsorted array:\n");
85     for (int i = 0; i < n; i++)
86     {
87         printf("%d, ", arr[i]);
88     }
89     return 0;
90 }

```

Terminal Output:

```

ab3> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3\"; if ($?) { gcc lab3-3.c -o lab3-3 }; if ($?) { .\lab3-3 }
● No of elements in array: 10
Enter the elements of the array:
32
32
5
7
9765
53
75
74
2
7
The elements of the array are:
32, 32, 5, 7, 9765, 53, 75, 74, 2, 7,
sorted array:
2, 5, 7, 7, 32, 32, 53, 74, 75, 9765,
PS c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3>

```

2. Set a threshold value of 5 for switching from Quick Sort to Insertion Sort when sorting small subarrays.
3. Implement Quick Sort to sort the array:
  - (a) If the size of the current subarray is greater than the threshold, continue with Quick Sort.
  - (b) If the size of the current subarray is less than or equal to the threshold, use Insertion Sort instead.
4. Sort the array in ascending order using this hybrid approach.
5. Display the sorted array after applying the hybrid sorting algorithm.

The image shows two screenshots of a C++ IDE (Visual Studio Code) implementing a hybrid sorting algorithm. The first screenshot shows the implementation of the insertion sort and partition functions. The second screenshot shows the implementation of the partition and quicksort functions. The terminal output shows the array being sorted and the final sorted array.

**First Screenshot: Insertion Sort and Partition Functions**

```

1 #include <stdio.h>
2
3 int insertion_sort(int arr[], int n)
4 {
5     for (int i = 1; i < n; i++)
6     {
7         int k = arr[i];
8         int j = i - 1;
9         while (j >= 0)
10        {
11            if (arr[j] > k)
12            {
13                arr[j + 1] = arr[j];
14                j--;
15            }
16            else
17            {
18                break;
19            }
20        }
21        arr[j + 1] = k;
22    }
23    return 0;
24 }
25
26 int partition(int arr[], int low, int high)
27 {
28     if (high - low + 1 <= 5)
29     {
30         insertion_sort(arr + low, high - low + 1);
31         return -1;
32     }
33     int p = arr[low];
34     int i = low;
35     int j = high;
36     while (i < j)
37     {

```

**Second Screenshot: Partition and Quicksort Functions**

```

38         while (arr[i] <= p && i < high)
39             i++;
40         while (arr[j] > p && j > low)
41             j--;
42         if (i < j)
43         {
44             int temp = arr[i];
45             arr[i] = arr[j];
46             arr[j] = temp;
47         }
48         int temp = arr[low];
49         arr[low] = arr[j];
50         arr[j] = temp;
51         return j;
52     }
53 }
54
55 int quicksort(int arr[], int low, int high)
56 {
57     if (low < high)
58     {
59         int p_index = partition(arr, low, high);
60         if (p_index != -1)
61         {
62             quicksort(arr, low, p_index - 1);
63             quicksort(arr, p_index + 1, high);
64         }
65     }
66     return 0;
67 }

```

**Terminal Output:**

```

PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3>
cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3\" ; if ($?) { gcc lab3-3.c -o lab3-3 } ; if ($?) { .\lab3-3 }
No of elements in array: 10
Enter the elements of the array:
32
32
5
7
9765
53
75
74
2
7
The elements of the array are:
32, 32, 5, 7, 9765, 53, 75, 74, 2, 7,
sorted array:
2, 5, 7, 7, 32, 32, 53, 74, 75, 9765,
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab3>

```