

Left subarray : left to mid

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
6	3	9	1	7	2	8	4	5

left mid right

Right Subarray : mid + 1 to Right

[0]	[1]	[2]	[3]	[4]
6	3	9	1	7

left mid right

[5]	[6]	[7]	[8]
2	8	4	5

Left mid right

[0]	[1]	[2]
6	3	9

Left mid right

[3]	[4]
1	7

Left right
mid

[5]	[6]
2	8

Left right
mid

[7]	[8]
4	5

Left right
mi

3 6

[0]	[1]
6	3

Left right
mid

1

[2]
9

Left
right

9

[3]
1

Left
right

[4]
7

Left
right

[5]
2

Left right
right

[6]
8

Left right

[7]
4

Left right

[8]
5

Left
right

[0]
6

Left
right

[1]
3

Left
right

Merge all the Small size Subarrays in ascending order in temp array

[0]	[1]
3	6

[0]	[1]
1	9

[0]	[1]
2	7

[0]	[1]
4	8

[0]
5

[0]	[1]	[2]	[3]
1	3	6	9

[0]	[1]	[2]	[3]
2	4	7	8

[0]
5

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
1	2	3	4	6	7	8	9

[0]
5

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
1	2	3	4	5	6	7	8	9

```

void merge_sort(int arr[SIZE],int
left,int right)
{
    int mid = (left + right) / 2;
    if(left >= right)
        return;
    merge_sort(arr,left,mid); //
left sub array
    merge_sort(arr,mid+1,right); //
right sub array
    // merge the sub array in a
sorted order in a temp array
    int temp_size = right-left + 1;
    int *ptr =
    malloc(sizeof(int)*temp_size);
    int i = left, j = mid+1, k = 0;
    while(i<= mid && j <= right)
    {
        // ptr[k++] = arr[i]
        < arr[j] ? arr[i++] : arr[j++];
        if(arr[i] < arr[j])
        {
            ptr[k] = arr[i];
            i++; k++;
        }
        else
        {
            ptr[k] = arr[j];
            j++; k++;
        }
    }
    while(i <= mid)
    {
        ptr[k++] = arr[i++];
    }
    while(j <= right)
    {
        ptr[k++] = arr[j++];
    }
    // copy the sorted data of temp
array into the main array
    for(int i = 0; i
    < temp_size;i++)
    {
        arr[left + i] = ptr[i]; //
5
    }
    free(ptr);
    ptr = NULL;
}

```