

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

Left subarray : left to mid

mic

right

Right Subarray : mid + 1 to Right

[0]	[1]	[2]	[3]	[4]
6	3	9	1	7
left		mid		right

left                          right

[5]	[6]	[7]	[8]
2	8	4	5
Left	mid		right

2	3	4	5
Left	mid	right	

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

Left mid right

[3]	[4]
1	7

Left right

[5]	[6]
2	8

Left right

[7]	[8]
4	5

Left right

3 6.

1

9.

id

-right

ft

```
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;
}
```