**ASSIGNMENT-7**

**1.PROBLEM STATEMENT**

Write a program in C to implement merge sort.

**2.ALGORITHMS**

Algorithm **Merge\_Sort**

**Input:**The pointer ‘arr’ to the array of integers, the left index ‘left’ and right index ‘right’ of the concerned sublist.

**Output:**The array ‘arr’ with the elements in sorted order.

**Remarks:**The function works recursively.

**Steps:**

1. **If**(left≥right) **then** // recursion termination condition
2. **Return**
3. **Else**
4. mid=(left+right)/2 //calculation of mid
5. **Merge\_Sort**(arr,left,mid) //Divide for the left subarray
6. **Merge\_Sort**(arr,mid+1,right) // Divide for the right subarray
7. **Merge**(arr,left,mid,right) // Merge the divided arrays in order

Algorithm **Merge**

**Input:** The pointer ‘arr’ to the array containing the sublists, the left index ‘left’ , the right index ‘right’ and the middle index ‘mid’.

**Output:** The elements of the two sublists merged back into the original array ‘arr’ in the right order.

**Remarks:**Merging happens using an auxillary array named ‘sorted’.

**Steps:**

1. i=left,j=mid+1,k=left //i and j hold the starting index of subarrays
2. **While**(i≤mid **AND** j≤right) **do** //while both arrays are not exhausted
3. **If**(arr[i]≤arr[j]) **then** //if left element ≤ right element
4. sorted[k]=arr[i] //insert left element in sorted array
5. i=i+1,k=k+1 //increment i and k by 1
6. **Else**
7. sorted[k]=arr[j] //insert right element in sorted array
8. j=j+1,k=k+1 //increment j and k by 1
9. **While**(i≤mid) **do** //if left sublist is remaining
10. sorted[k]=arr[i] //copy all elements to sorted array
11. k=k+1,i=i+1 //increment k and i
12. **While**(j≤right) do //if right sublist remains
13. sorted[k]=arr[j] //copy all elements to sorted array
14. k=k+1,j=j+1 //increment k and j
15. **For** i=left to right **do** //copying sorted array into input array
16. arr[i]=sorted[i]

**3.SOURCE CODE**

#include<stdlib.h>

#include<stdio.h>

void getarr(int \*arr,int size)

{

int i;

for(i=0;i<size;i++)

scanf("%d",&arr[i]);

}

void disparr(int \*arr,int size)

{

int i;

for(i=0;i<size;i++)

printf("%d ",arr[i]);

}

//function to merge two subarrays of a single array

void merge(int \*arr,int left,int mid,int right)

{

int sorted[20],i,j,k=left;

i=left; //starting index of the left subarray

j=mid+1; //starting index of the right subarray

while(i<=mid && j<=right) //while both arrays are not exhausted

{

if(arr[i]<=arr[j]) //if left subarray element is greater

{

sorted[k]=arr[i]; //put the element in sorted array

k++;i++;

}

else

{

sorted[k]=arr[j]; //otherwise put the right element

k++;j++;

}

}

while(i<=mid) //if left subarray is remaining

{

sorted[k]=arr[i]; //copy all remaining elements

k++;i++;

}

while(j<=right) //if right subarray is remaining

{

sorted[k]=arr[j]; //copy all remaining elements

k++;j++;

}

for(i=left;i<=right;i++) //copy sorted array into original array

arr[i]=sorted[i];

}

//recursive function to divide the array into subarrays

void mergesort(int \*arr,int left,int right)

{

int mid;

if(right<=left) //recursion termination condition

return;

else

{

mid=(left+right)/2; //calculation of middle index

mergesort(arr,left,mid); // divide for the left subarray

mergesort(arr,mid+1,right); // divide for the right subarray

merge(arr,left,mid,right); // merge the divided arrays

}

}

int main(void)

{

int \*arr,size,left=0,right;

printf("To sort a list of integers using Merge Sort: \n");

printf("Enter the number of elements needed: ");

scanf("%d",&size);

if(size<1) //input validation

{

printf("Invalid array size, please try again");

exit(1);

}

right=(size-1);

arr=(int\*)malloc(size\*sizeof(int)); //creating array in heap

printf("Enter %d elements of the array: ",size);

getarr(arr,size);

printf("Entered array: ");

disparr(arr,size);

mergesort(arr,left,right);

printf("\nSorted Array: ");

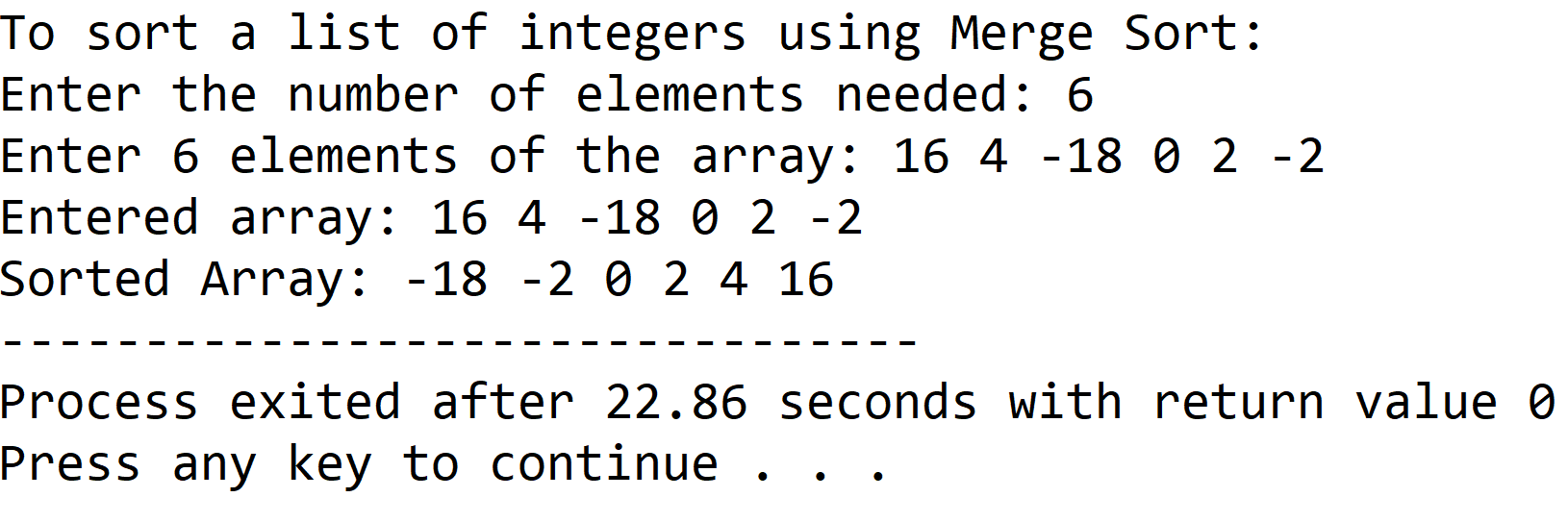
disparr(arr,size);

return 0;

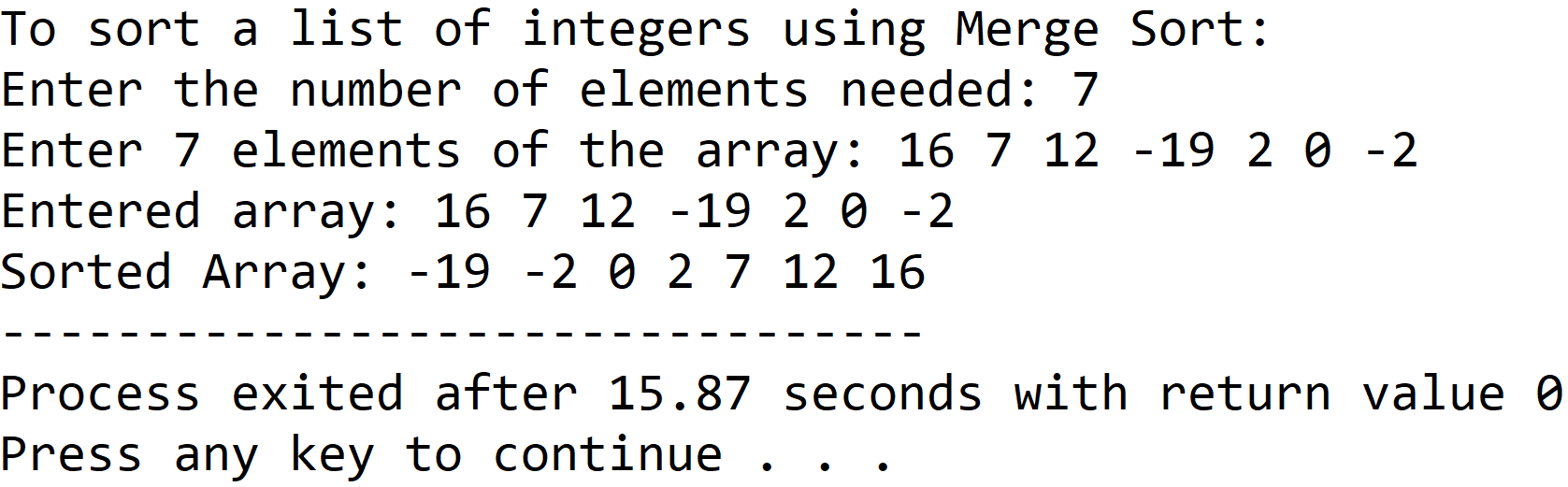
}

**4.OUTPUT**

**SET 1:** Even sized array



**SET 2:** Odd sized array



**5.DISCUSSIONS**

**Variable Description:**

* **\*arr:** pointer to an array
* **size:** the input length of the array
* **left:** lowermost index of the concerned array
* **right:** uppermost index of the concerned array
* **mid:** the index of the middle element of the concerned array
* **sorted**: auxillary array to perform merge procedure
* **i,j,k**: loop counters

**Limitations:**

* An integer array has been used to hold the list of integers, since arrays are static data structures, their size cannot be manipulated once they are constructed in the computer’s memory, also contiguous memory locations are necessary in order to create an array which is another drawback.

**Uses:**

* The program can be used to sort any list of integers in ascending order.For example, it can be used by educational institutions to arrange a random list of students on the basis of ascending order of their roll numbers.

**Future Scope:**

* The program can be made more memory efficient by replacing the array with a linked list.