**ASSIGNMENT - 6**

**PROBLEM STATEMENT**

Write a program in C to implement quick sort.

**ALGORITHMS**

Algorithm **Quick\_Sort**

**Input:** The pointer ‘arr’ to the array holding the list of integers, the left boundary index ‘left’ and and the right boundary index ‘right’.

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

**Remarks:** The algorithm works recursively.

**Steps:**

1. **If**(left<right) **then**
2. loc=**Partition**(arr,left,right) //get pivot element index
3. **Quick\_Sort**(arr,left,loc-1) //sort left subarray
4. **Quick\_Sort**(arr,loc+1,right) //sort right subarray
5. **EndIf**
6. **Stop**

Algorithm **Partition**

**Input:**The pointer ‘arr’ to the integer array, the left boundary index ‘left’ and the right boundary index ‘right’.

**Output:**The index of the pivot element placed in its sorted position.

**Steps:**

1. loc=left
2. **While**(left<right) **do**
3. **While**(arr[right]≥arr[loc] **AND** right>loc) **do**
4. right=right-1
5. **EndWhile**
6. **If**(arr[loc]>arr[right]) **then**
7. **Swap**(arr[loc],arr[right])
8. loc=right
9. left=left+1
10. **EndIf**
11. **While**(arr[left]≤arr[loc] **AND** left<loc) **do**
12. left=left+1
13. **EndWhile**
14. **If**(arr[loc]<arr[left]) **then**
15. **Swap**(arr[loc],arr[left])
16. loc=left
17. right=right-1
18. **EndIf**

**19. EndWhile**

**20.** **Return** loc

**21. Stop**

Algorithm **Swap**

**Input:** The two variables named ‘a’ and ‘b’ whose values are to be swapped.

**Output:** The values of variables ‘a’ and ‘b’ interchanged with one another.

**Remarks:** The variables must be passed as pointers.

**Steps:**

1. **a = a + b**
2. **b = a – b**
3. **a = a – b**
4. **Stop**

**3. SOURCE CODE**

#include<stdio.h>

#include<stdlib.h>

//function to take input in an array

void getarr(int \*arr,int size)

{

int i;

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

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

}

//function to display an array

void disparr(int \*arr,int size)

{

int i;

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

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

}

//function to swap two variables

void swap(int \*a,int\*b)

{

\*a=\*a+\*b;

\*b=\*a-\*b;

\*a=\*a-\*b;

}

//function to divide the array using pivot element

int partition(int \*arr,int left,int right)

{

int loc=left; //selecting left as pivot element

while(left<right) //traversing the whole array

{

//while right element is greater or equal to pivot element

while(arr[right]>=arr[loc] && loc<right)

right--;//decrement right

//if right element is less than pivot element

if(arr[right]<arr[loc])

{

swap(&arr[loc],&arr[right]);//swap right and pivot elements

loc=right;//set pivot location to right

left++;//left to right scan starts from left+1

}

//while left element is smaller/equal to pivot element

while(arr[left]<=arr[loc] && left<loc )

left++;//increment left

//if left element is greater than pivot element

if(arr[left]>arr[loc])

{

swap(&arr[left],&arr[loc]);//swap left and pivot element

loc=left;//set pivot location to left

right--;//right to left scan starts from right-1

}

}

return loc;

}

//recursive function for performing quicksort

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

{

int loc;

if(left<right)//while there are more than one element

{

loc=partition(arr,left,right);//find pivot location

quicksort(arr,left,loc-1);//sort left subarray

quicksort(arr,loc+1,right);//sort right subarray

}

}

//function for input validation

void validate(int size)

{

if(size<2)

{

printf("The size must be atleast two");

exit(1);

}

}

int main(void)

{

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

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

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

scanf("%d",&size);

validate(size);//validating input

right=size-1;

arr=(int\*)malloc(size\*sizeof(int));

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

getarr(arr,size);

printf("Entered array: ");

disparr(arr,size);

quicksort(arr,left,right);

printf("\nSorted array: ");

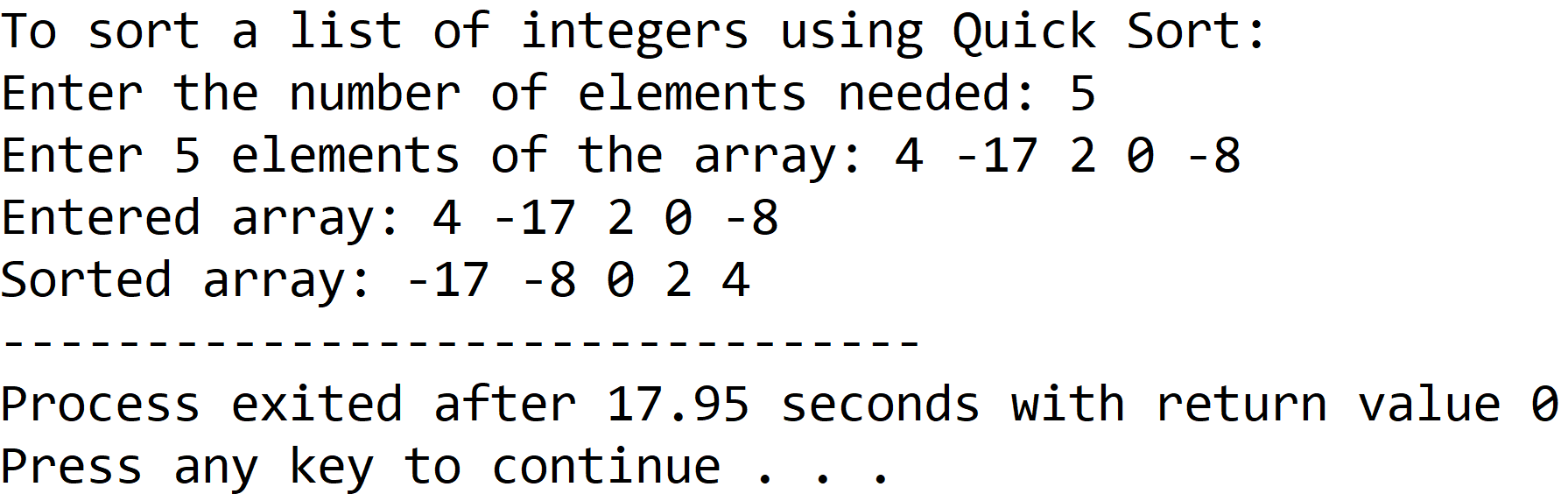
disparr(arr,size);

return 0;

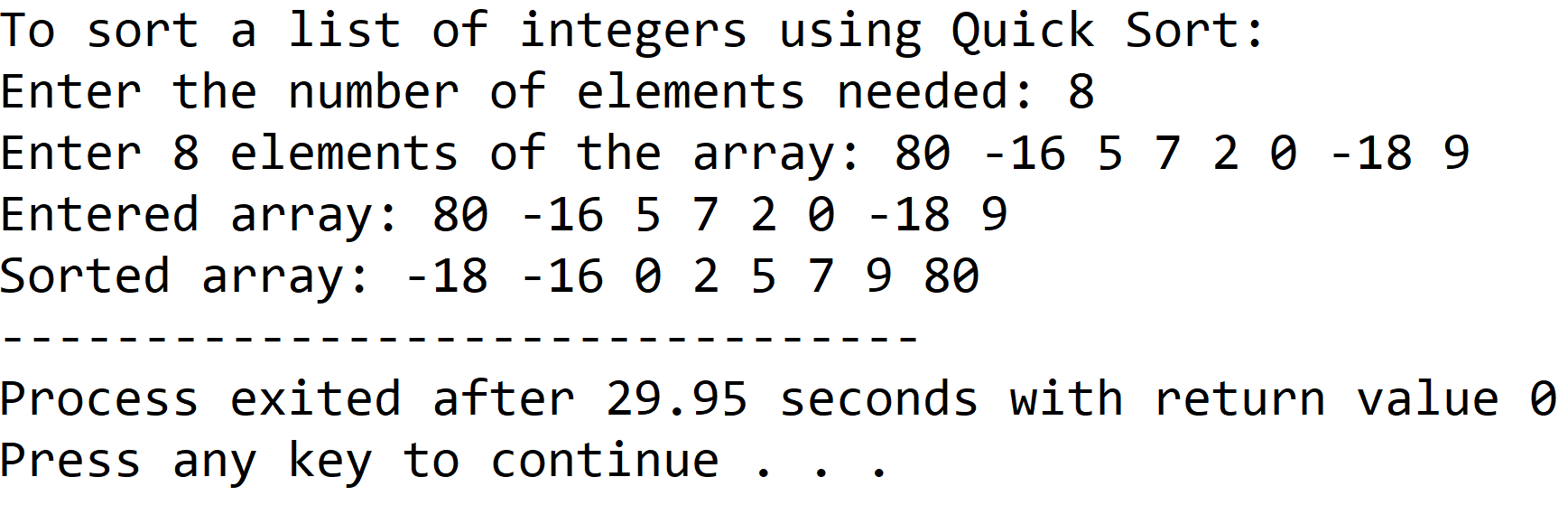
}

**4.OUTPUT**

**SET 1:** Odd length input array



**SET 2:** Even length input array



**5.DISCUSSIONS**

**Variable Description**

* **\*arr:** pointer to an integer array.
* **size:** length of the array input by the user.
* **left:** lowermost boundary index of concerned sublist.
* **right:** uppermost boundary index of concerned sublist.
* **loc:** index of the pivot element.
* **\*a,\*b:** pointers for swapping procedure.
* **i:** loop counter.

**Limitations**

* The program uses an integer array, which is a static data structure whose size cannot be manipulated once it is created in the computer’s memory, contiguous memory locations are needed to construct an array and if contiguous memory locations are not available, the array fails to be created.

**Uses**

* The program can be used to sort any list of integers in ascending order. For example, it can be used to sort files in a database based upon their order of creation.

**Future Scope**

* The array used in the program can be replaced with a linked list, making the program more memory efficient.

**Teacher’s Signature**