Experiment No.2

Title : Sorting on Array

Register No.24141028

Program1:

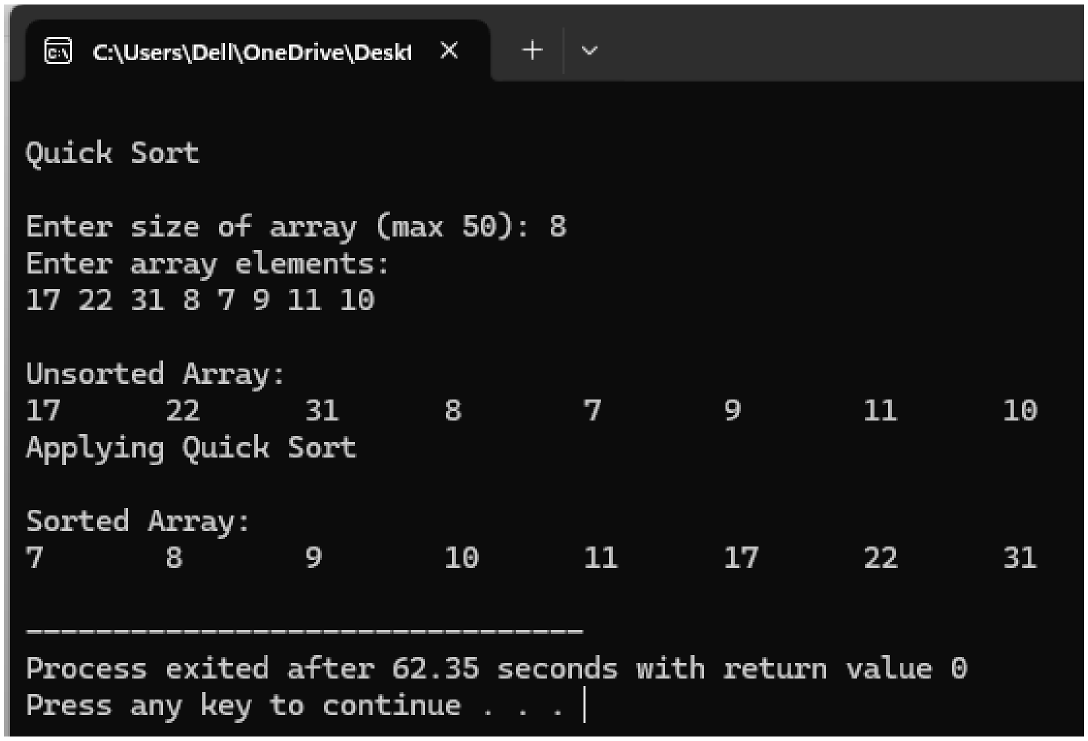
Quick Sort -

```
#include <stdio.h>

#include <stdlib.h>

#define SIZE 50

void swap(int *x, int *y)

{

        int temp = *x;

        *x = *y;

        *y = temp;

}


int partition(int arr[], int low, int high) {

        int pivot = arr[low];

        int start = low;

        int end = high;
while (start < end)
{
        while (arr[start] <= pivot)
            start++;
        while (arr[end] > pivot)
            end--;
        if (start < end)
            swap(&arr[start], &arr[end]);
    }
   swap(&arr[low], &arr[end]);
    return end;
}
void quick_sort(int arr[], int low, int high) {
    if (low < high) {
        int loc = partition(arr, low, high);
        quick_sort(arr, low, loc - 1);
        quick_sort(arr, loc + 1, high);
    }
}
```

```
int main()
 {
    int arr[SIZE], n, i;
    printf("¥nQuick Sort¥n¥n");
    printf("Enter size of array (max %d): ", SIZE);
    scanf("%d", &n);
    printf("Enter array elements:¥n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("¥nUnsorted Array:¥n");
    for (i = 0; i < n; i++) {
        printf("%d¥t", arr[i]);
    }
    printf("¥nApplying Quick Sort");
    quick_sort(arr, 0, n - 1);
    printf("¥n¥nSorted Array:¥n");
    for (i = 0; i < n; i++) {
        printf("%d¥t", arr[i]);
    }
```

```
        printf("¥n");
    return 0;
}
```

Output:



```
C:\Users\Dell\OneDrive\Deskt

Quick Sort

Enter size of array (max 50): 8
Enter array elements:
17 22 31 8 7 9 11 10

Unsorted Array:
17      22      31      8       7       9       11      10
Applying Quick Sort

Sorted Array:
7       8       9       10      11      17      22      31

--------------------------------
Process exited after 62.35 seconds with return value 0
Press any key to continue . . .
```

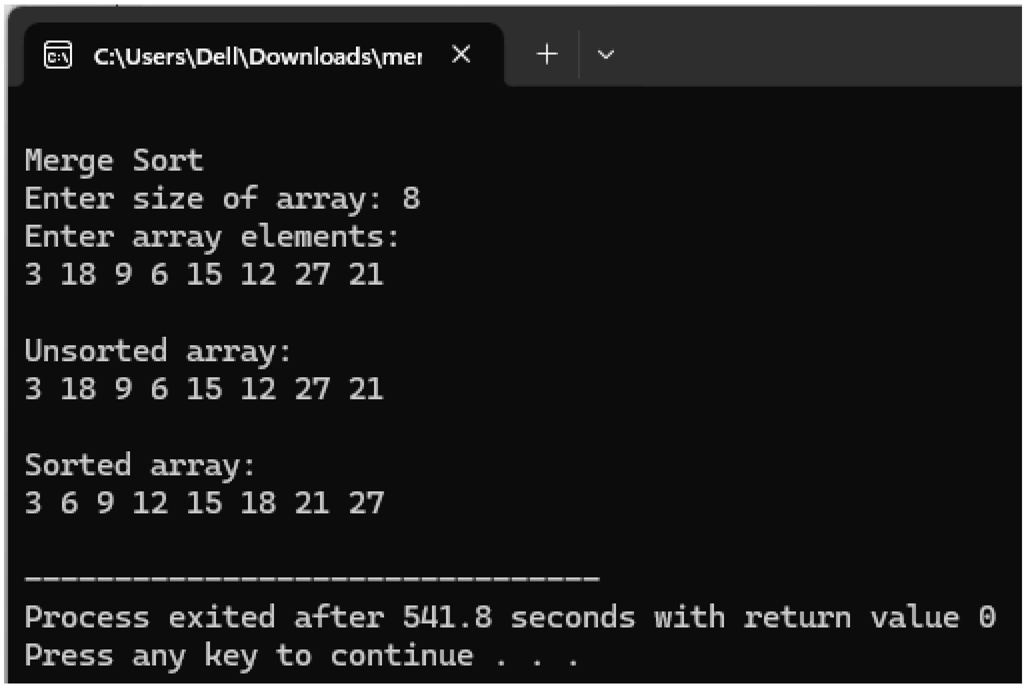
Program2:

Merge Sort -

```
#include <stdio.h>
#define SIZE 50
void merge(int arr[], int lb, int mid, int ub)
{
        int brr[SIZE];
        int i = lb;

        int j = mid + 1;

        int k = lb;

while (i <= mid && j <= ub)

{

                if (arr[i] < arr[j])

{

                        brr[k++] = arr[i++];

  }

  else

{

                brr[k++] = arr[j++];

}

        }

        while (i <= mid)

  {

                brr[k++] = arr[i++];

  }

        while (j <= ub)

  {

                brr[k++] = arr[j++];

  }
```

```
        for (k = lb; k <= ub; k++)
        {
                arr[k] = brr[k];
        }
}

void mergeSort(int arr[], int lb, int ub)
  {
        if (lb < ub)
{
                int mid = (lb + ub) / 2;
                mergeSort(arr, lb, mid);
                mergeSort(arr, mid + 1, ub);
                merge(arr, lb, mid, ub);
        }
}

int main()
  {
        int arr[SIZE], n;
        int i;
        printf("¥nMerge Sort¥n");
        printf("Enter size of array: ");
        scanf("%d", &n);
                if (n > SIZE)
    {
      printf("Array size exceeds maximum allowed size (%d).¥n", SIZE);
                return 1;
        }
        printf("Enter array elements:¥n");
        for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        }
        printf("¥nUnsorted array:¥n");
        for(i = 0; i < n; i++)
     {
                printf("%d ", arr[i]);
        }
        mergeSort(arr, 0, n - 1);
        printf("¥n¥nSorted array:¥n");
        for (i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        printf("¥n");
    return 0;
}
```

Output:



```
Merge Sort
Enter size of array: 8
Enter array elements:
3 18 9 6 15 12 27 21

Unsorted array:
3 18 9 6 15 12 27 21

Sorted array:
3 6 9 12 15 18 21 27

--------------------------------
Process exited after 541.8 seconds with return value 0
Press any key to continue . . .
```