Name: Badal Prabhakar Wanjari

Branch: Computer Technology

Section: B

Roll No. 140

Reg No. 20011045

Subject: Data Structures

Practical – 6

Aim: Program for allocating memory dynamically for sing dimensional array and sort it using quick sort and merge sort.

Program:

Sorting Array using quicksort.

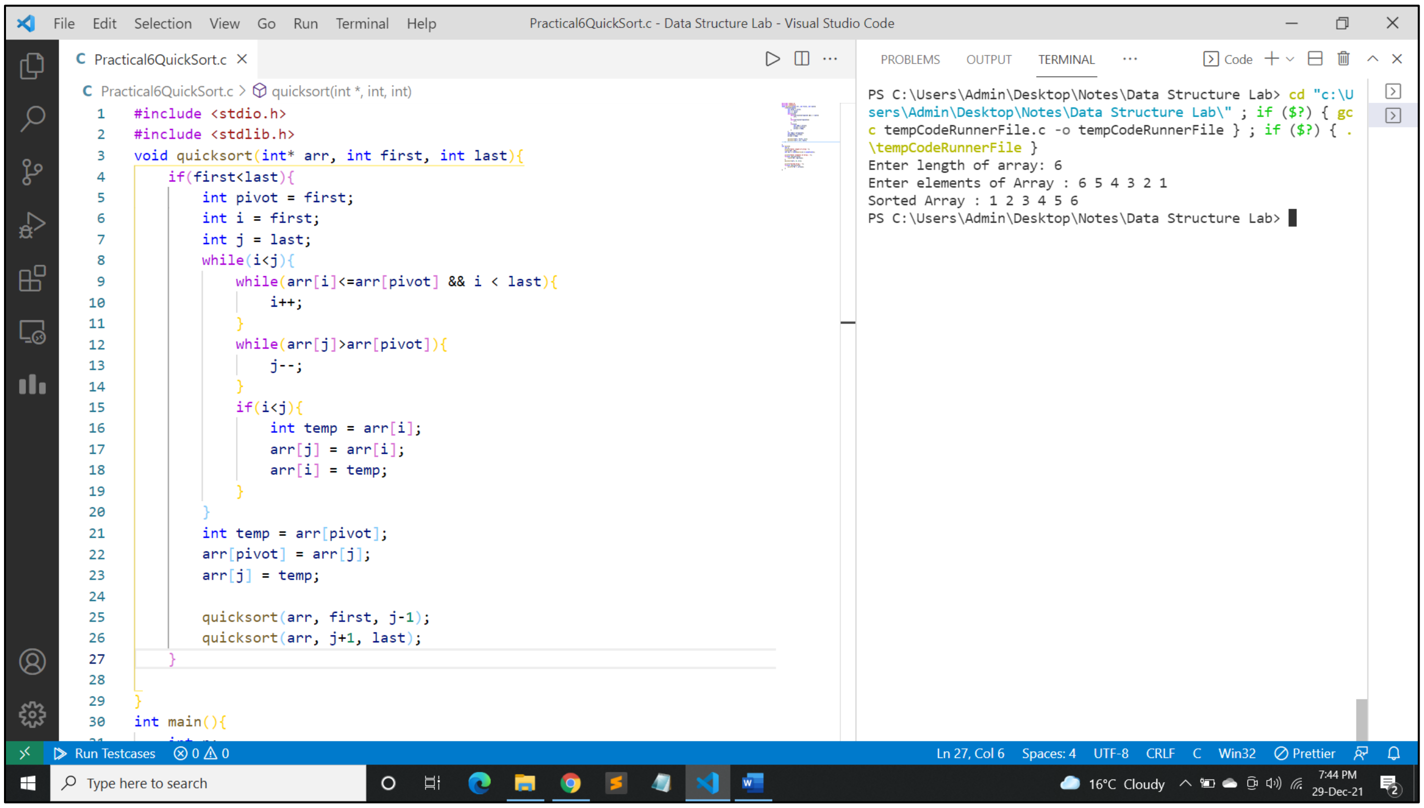
```
#include <stdio.h>
#include <stdlib.h>
void quicksort(int* arr, int first, int last){
    if(first<last){
        int pivot = first;
        int i = first;
        int j = last;
        while(i<j){
            while(arr[i]<=arr[pivot] && i < last){
                i++;
            }
            while(arr[j]>arr[pivot]){
                j--;
            }
            if(i<j){
                int temp = arr[i];
                arr[j] = arr[i];
                arr[i] = temp;
            }
        }
        int temp = arr[pivot];
        arr[pivot] = arr[j];
        arr[j] = temp;

        quicksort(arr, first, j-1);
        quicksort(arr, j+1, last);
    }

}
int main(){
    int n;
```

```c
    printf("Enter length of array: ");
    scanf("%d", &n);
    int *arr = (int*)malloc(n * sizeof(int));

    printf("Enter elements of Array : ");
    for(int i=0;i<n;i++){
        scanf("%d", &arr[i]);
    }
    quicksort(arr, 0, n-1);

    printf("Sorted Array : ");
    for(int i=0;i<n;i++){
        printf("%d ", arr[i]);
    }
}
```

Output:

```
Enter length of array: 6
Enter elements of Array : 6 5 4 3 2 1
Sorted Array : 1 2 3 4 5 6
```

Screenshot:



```c
#include <stdio.h>
#include <stdlib.h>
void quicksort(int* arr, int first, int last){
    if(first<last){
        int pivot = first;
        int i = first;
        int j = last;
        while(i<j){
            while(arr[i]<=arr[pivot] && i < last){
                i++;
            }
            while(arr[j]>arr[pivot]){
                j--;
            }
            if(i<j){
                int temp = arr[i];
                arr[j] = arr[i];
                arr[i] = temp;
            }
        }
        int temp = arr[pivot];
        arr[pivot] = arr[j];
        arr[j] = temp;

        quicksort(arr, first, j-1);
        quicksort(arr, j+1, last);
    }

}
int main(){
```

```
PS C:\Users\Admin\Desktop\Notes\Data Structure Lab> cd "c:\U
sers\Admin\Desktop\Notes\Data Structure Lab\" ; if ($?) { gc
c tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .
\tempCodeRunnerFile }
Enter length of array: 6
Enter elements of Array : 6 5 4 3 2 1
Sorted Array : 1 2 3 4 5 6
PS C:\Users\Admin\Desktop\Notes\Data Structure Lab>
```

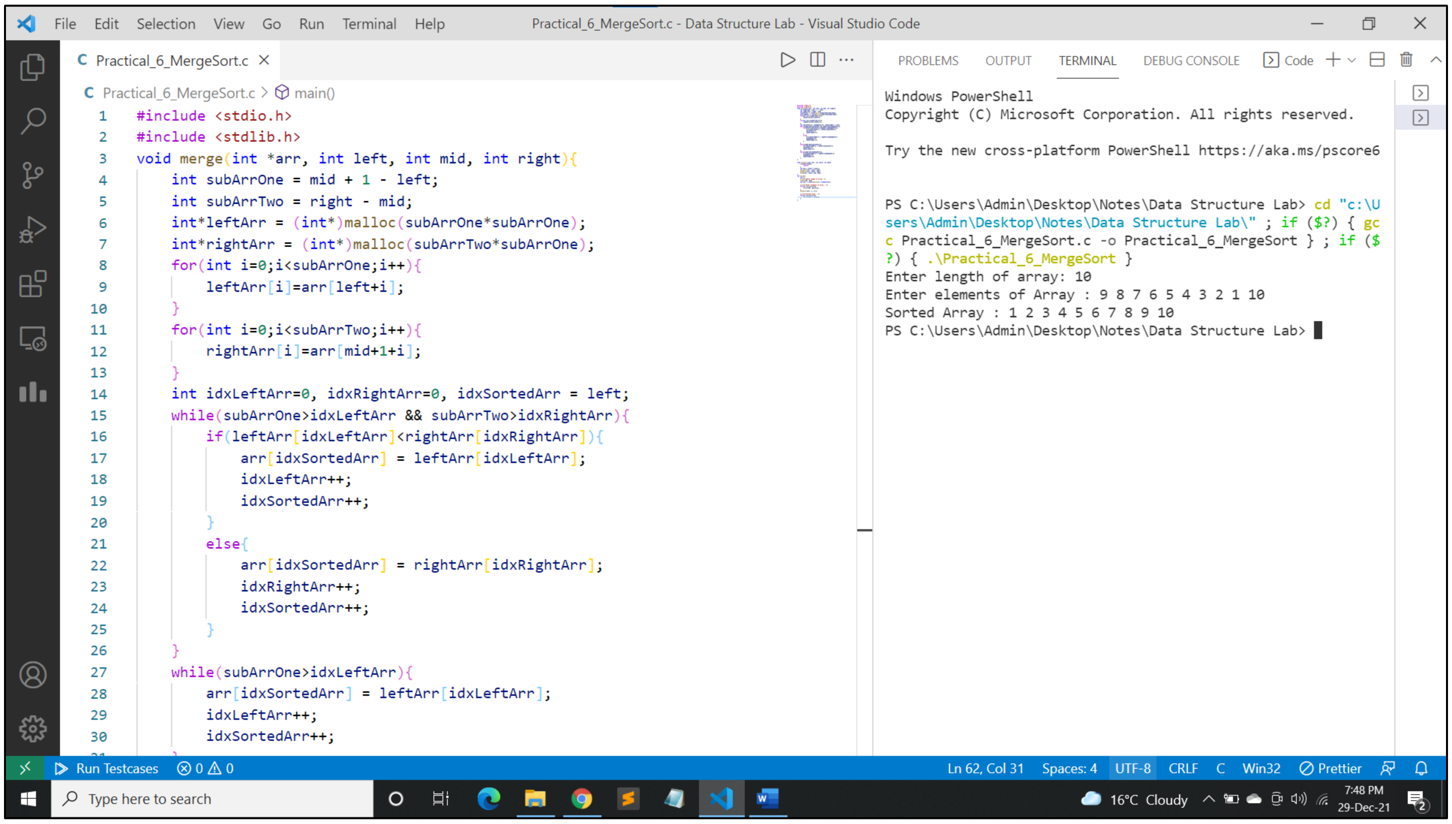
Sorting Array using mergesort.

```c
#include <stdio.h>
#include <stdlib.h>
void merge(int *arr, int left, int mid, int right){
    int subArrOne = mid + 1 - left;
    int subArrTwo = right - mid;
    int*leftArr = (int*)malloc(subArrOne*subArrOne);
    int*rightArr = (int*)malloc(subArrTwo*subArrOne);
    for(int i=0;i<subArrOne;i++){
        leftArr[i]=arr[left+i];
    }
    for(int i=0;i<subArrTwo;i++){
        rightArr[i]=arr[mid+1+i];
    }
    int idxLeftArr=0, idxRightArr=0, idxSortedArr = left;
    while(subArrOne>idxLeftArr && subArrTwo>idxRightArr){
        if(leftArr[idxLeftArr]<rightArr[idxRightArr]){
            arr[idxSortedArr] = leftArr[idxLeftArr];
            idxLeftArr++;
            idxSortedArr++;
        }
        else{
            arr[idxSortedArr] = rightArr[idxRightArr];
            idxRightArr++;
            idxSortedArr++;
        }
    }
    while(subArrOne>idxLeftArr){
        arr[idxSortedArr] = leftArr[idxLeftArr];
        idxLeftArr++;
        idxSortedArr++;
    }
    while(subArrTwo>idxRightArr){
        arr[idxSortedArr] = rightArr[idxRightArr];
        idxRightArr++;
        idxSortedArr++;
    }
}

void mergesort(int *arr, int start, int end){
    if(start>=end){
        return;
    }
    int mid = (start + end)/2;
    mergesort(arr, start, mid);
    mergesort(arr, mid+1, end);
    merge(arr, start, mid, end);
}
int main(){
    int n;
    printf("Enter length of array: ");
    scanf("%d", &n);
    int *arr = (int*)malloc(n * sizeof(int));

    printf("Enter elements of Array : ");
    for(int i=0;i<n;i++){
        scanf("%d", &arr[i]);
    }
    mergesort(arr, 0, n-1);

    printf("Sorted Array : ");
    for(int i=0;i<n;i++){
        printf("%d ", arr[i]);
    }
}
```

Output:

```
Enter length of array: 10
Enter elements of Array : 9 8 7 6 5 4 3 2 1 10
Sorted Array : 1 2 3 4 5 6 7 8 9 10
```

Screenshot:



```c
#include <stdio.h>
#include <stdlib.h>
void merge(int *arr, int left, int mid, int right){
    int subArrOne = mid + 1 - left;
    int subArrTwo = right - mid;
    int*leftArr = (int*)malloc(subArrOne*subArrOne);
    int*rightArr = (int*)malloc(subArrTwo*subArrOne);
    for(int i=0;i<subArrOne;i++){
        leftArr[i]=arr[left+i];
    }
    for(int i=0;i<subArrTwo;i++){
        rightArr[i]=arr[mid+1+i];
    }
    int idxLeftArr=0, idxRightArr=0, idxSortedArr = left;
    while(subArrOne>idxLeftArr && subArrTwo>idxRightArr){
        if(leftArr[idxLeftArr]<rightArr[idxRightArr]){
            arr[idxSortedArr] = leftArr[idxLeftArr];
            idxLeftArr++;
            idxSortedArr++;
        }
        else{
            arr[idxSortedArr] = rightArr[idxRightArr];
            idxRightArr++;
            idxSortedArr++;
        }
    }
    while(subArrOne>idxLeftArr){
        arr[idxSortedArr] = leftArr[idxLeftArr];
        idxLeftArr++;
        idxSortedArr++;
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6


PS C:\Users\Admin\Desktop\Notes\Data Structure Lab> cd "c:\U
sers\Admin\Desktop\Notes\Data Structure Lab\" ; if ($?) { gc
c Practical_6_MergeSort.c -o Practical_6_MergeSort } ; if ($
?) { .\Practical_6_MergeSort }
Enter length of array: 10
Enter elements of Array : 9 8 7 6 5 4 3 2 1 10
Sorted Array : 1 2 3 4 5 6 7 8 9 10
PS C:\Users\Admin\Desktop\Notes\Data Structure Lab>
```

Conclusion: I have successfully completed practical 6.