Q1. Implement Bubble Sort

Ans.

#include <stdio.h>

#include <stdlib.h>

void bubble\_sort(int arr[], int len){

    int temp;

    int swapped = 0;

    for(int i=0; i<len; i++){

        swapped = 0;

        for (int j=0; j<len-i-1; j++){

            if(arr[j]>arr[j+1]){

                swapped = 1;

                temp = arr[j];

                arr[j] = arr[j+1];

                arr[j+1] = temp;

            }

        }

        if(swapped == 0){

            break;

        }

    }

}

int main()

{

    int len, val;

    printf("Enter length of array: ");

    scanf("%d", &len);

    if(len<=0){

        printf("\nArray length must be greater than 0!");

        return EXIT\_SUCCESS;

    }

    int arr[len];

    printf("\nEnter array values:\n");

    for(int i=0; i<len; i++){

        scanf(" %d", &val);

        arr[i] = val;

    }

    bubble\_sort(arr, len);

    printf("\nSorted array: ");

    for(int i=0; i<len; i++){

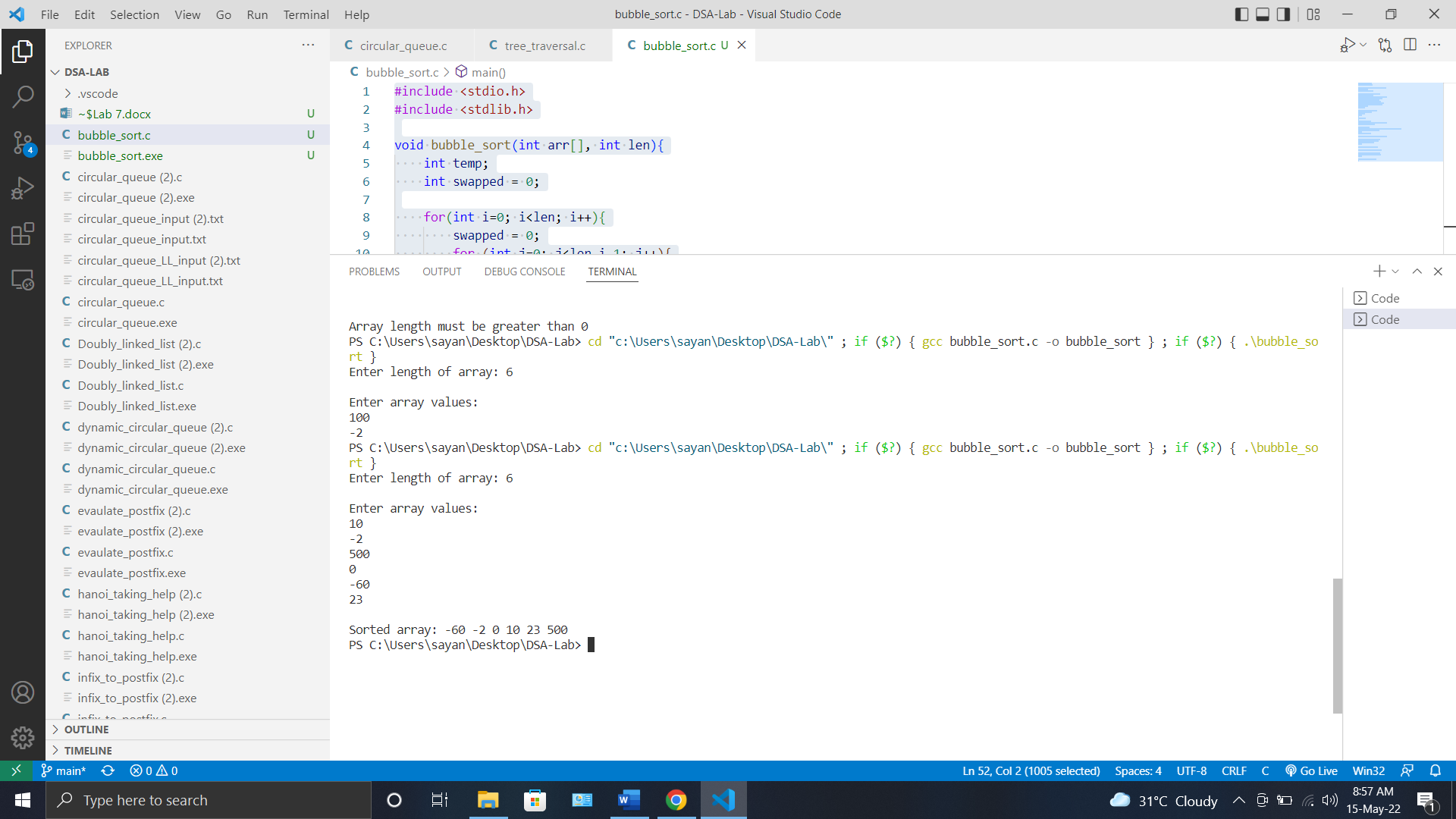
        printf("%d ", arr[i]);

    }

    return EXIT\_SUCCESS;

}

Output



Q2. Implement Quick Sort.

Ans.

#include <stdio.h>

#include <stdlib.h>

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

    int temp, index = low;

    low = low+1;

    while(low<=high){

        if(arr[low]<arr[index]){

            temp = arr[index];

            arr[index] = arr[low];

            arr[low] = arr[index+1];

            arr[index+1] = temp;

            index++;

        }

        low++;

    }

    return index;

}

void quick\_sort(int arr[], int low, int high){

    int index = partition(arr, low, high);

    if(low<=index-1){

        quick\_sort(arr, low, index-1);

    }

    if(index+1<=high){

        quick\_sort(arr, index+1, high);

    }

}

int main()

{

    int len, val;

    printf("Enter length of array: ");

    scanf("%d", &len);

    if(len<=0){

        printf("\nArray length must be greater than 0!");

        return EXIT\_SUCCESS;

    }

    int arr[len];

    printf("\nEnter array values:\n");

    for(int i=0; i<len; i++){

        scanf(" %d", &val);

        arr[i] = val;

    }

    quick\_sort(arr, 0, len-1);

    printf("\nSorted array: ");

    for(int i=0; i<len; i++){

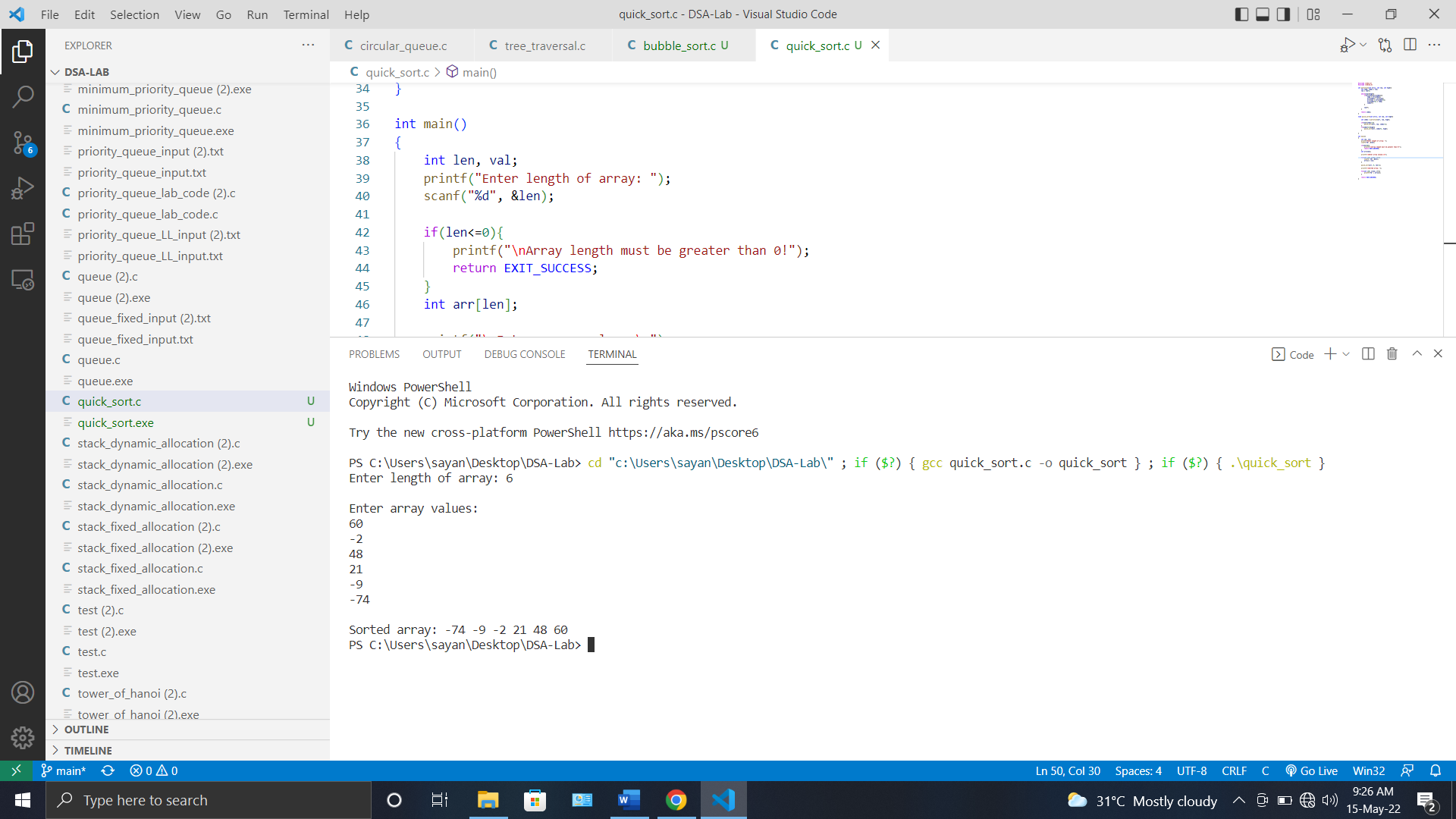
        printf("%d ", arr[i]);

    }

    return EXIT\_SUCCESS;

}

Output



Q3. Implement Merge Sort

Ans.

#include <stdio.h>

#include <stdlib.h>

void merge(int arr[], int l, int m, int r){

    int p\_len = m-l+1;

    int s\_len = r-m;

    int p\_arr[p\_len];

    int s\_arr[s\_len];

    int i, j;

    for(i=0; i<p\_len; i++){

        p\_arr[i] = arr[l+i];

    }

    for(i=0; i<s\_len; i++){

        s\_arr[i] = arr[m+1+i];

    }

    i=0;

    j=0;

    while(i<p\_len && j<s\_len){

        if(s\_arr[j]<p\_arr[i]){

            arr[l] = s\_arr[j];

            j++;

        }

        else{

            arr[l] = p\_arr[i];

            i++;

        }

        l++;

    }

    while (i<p\_len)

    {

        arr[l] = p\_arr[i];

        i++;

        l++;

    }

    while (j<s\_len)

    {

        arr[l] = s\_arr[j];

        j++;

        l++;

    }

}

void merge\_sort(int arr[], int low, int high){

    if(low<high){

        int middle = low + (high-low)/2;

        merge\_sort(arr, low, middle);

        merge\_sort(arr, middle+1, high);

        merge(arr, low, middle, high);

    }

}

int main()

{

    int len, val;

    printf("Enter length of array: ");

    scanf("%d", &len);

    if(len<=0){

        printf("\nArray length must be greater than 0!");

        return EXIT\_SUCCESS;

    }

    int arr[len];

    printf("\nEnter array values:\n");

    for(int i=0; i<len; i++){

        scanf(" %d", &val);

        arr[i] = val;

    }

    merge\_sort(arr, 0, len-1);

    printf("\nSorted array: ");

    for(int i=0; i<len; i++){

        printf("%d ", arr[i]);

    }

    return EXIT\_SUCCESS;

}

Output

