# ASSINGMENT-1(DSA)

# Animesh Maiti

# 2130006

Q1. *// WARP (Write a Recursive Program) to search an element in a dynamic array of n integers using linear search.*

#include <stdio.h>

#include <stdlib.h>

int searchElement(int arr[], int size, int element)

{

    for (int i = 0; i < size; i++)

    {

        if (arr[i] == element)

        {

            printf("element found at indx %d",i);

            return i;

        }

    }

    printf("element not found");

}

int main()

{

    int \*arr, n=5,element;

    printf("animesh maiti 2130006");

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

    printf("enter 5 elements\n");

    for (int i = 0; i < n; i++)

    {

        scanf(" %d",&arr[i]);

    }

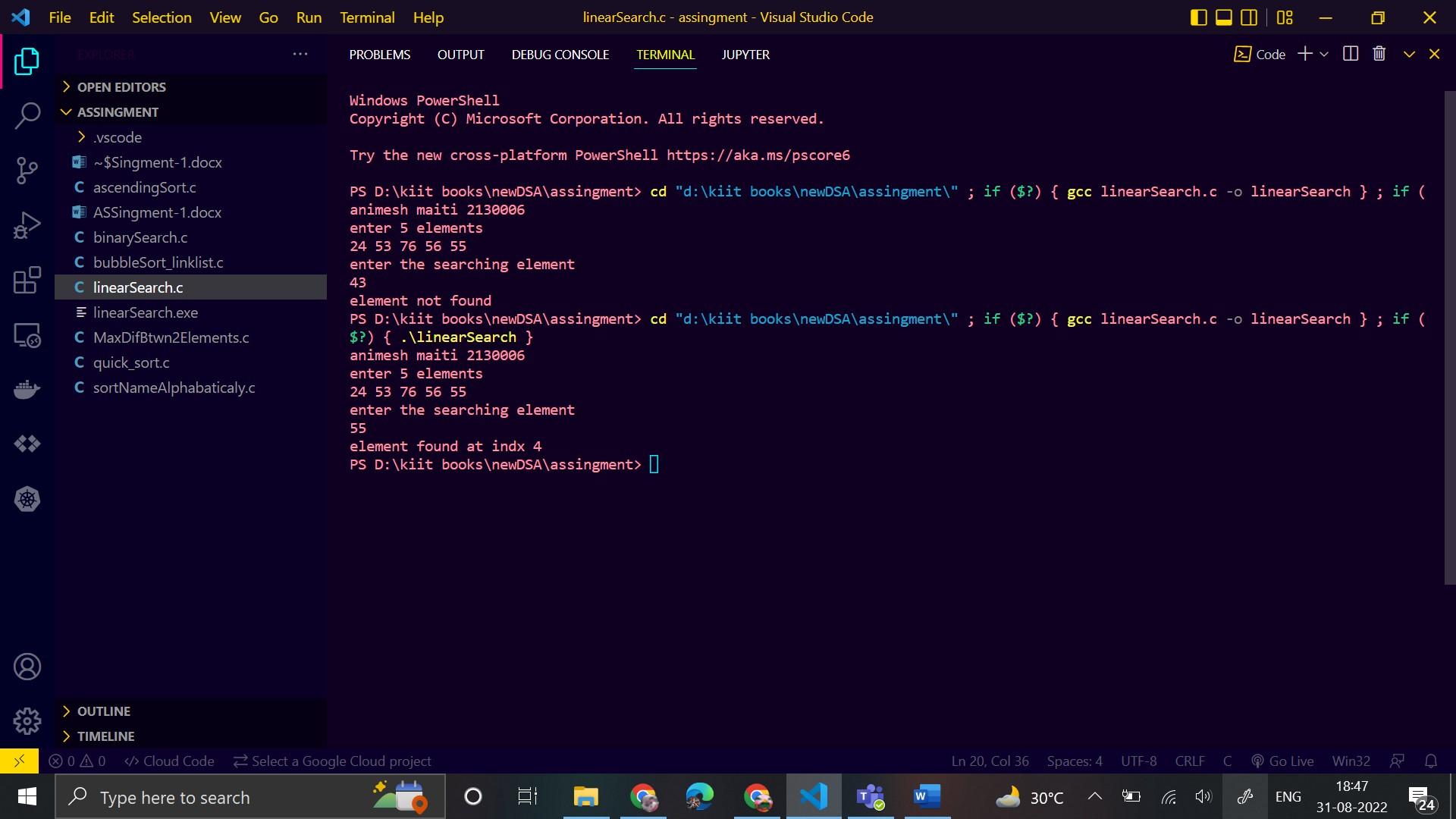
    printf("enter the searching element\n");

    scanf("%d",&element);

    searchElement(arr,n,element);

    return 0;

}



Q2. WARP using recursion to search an element in a dynamic array of n integers using binary search.

*// WARP using recursion to search an element in a dynamic array of n integers using binary search.*

#include <stdio.h>

#include <stdlib.h>

void bubbleSort(int arr[], int n)

{

    int i, j, temp;

    for (i = 0; i < n - 1; i++)

    {

        for (j = 0; j < n - i - 1; j++)

        {

            if (arr[j] > arr[j + 1]) *// condition for swaping*

            {

*// swap element*

                temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

            }

        }

    }

}

int binarySearch(int arr[], int low, int high, int element)

{

    int mid;

    if (high >= low)

    {

        mid = (low + high) / 2;

        if (arr[mid] == element)

        {

            return mid + 1;

        }

        else if (arr[mid] < element)

        {

            return binarySearch(arr, mid + 1, high, element);

        }

        else

        {

            return binarySearch(arr, low, mid - 1, element);

        }

    }

    return -1;

}

int main()

{

    printf("animesh maiti 2130006\n");

    int \*arr, n = 5, element,res;

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

    printf("enter 5 elements\n");

    for (int i = 0; i < n; i++)

    {

        scanf(" %d", &arr[i]);

    }

    printf("enter the searching element\n");

    scanf("%d", &element);

    bubbleSort(arr,n);

    res= binarySearch(arr,0,n-1,element);

    if (res==-1)

    {

        printf("element is not found\n");

    }

    else

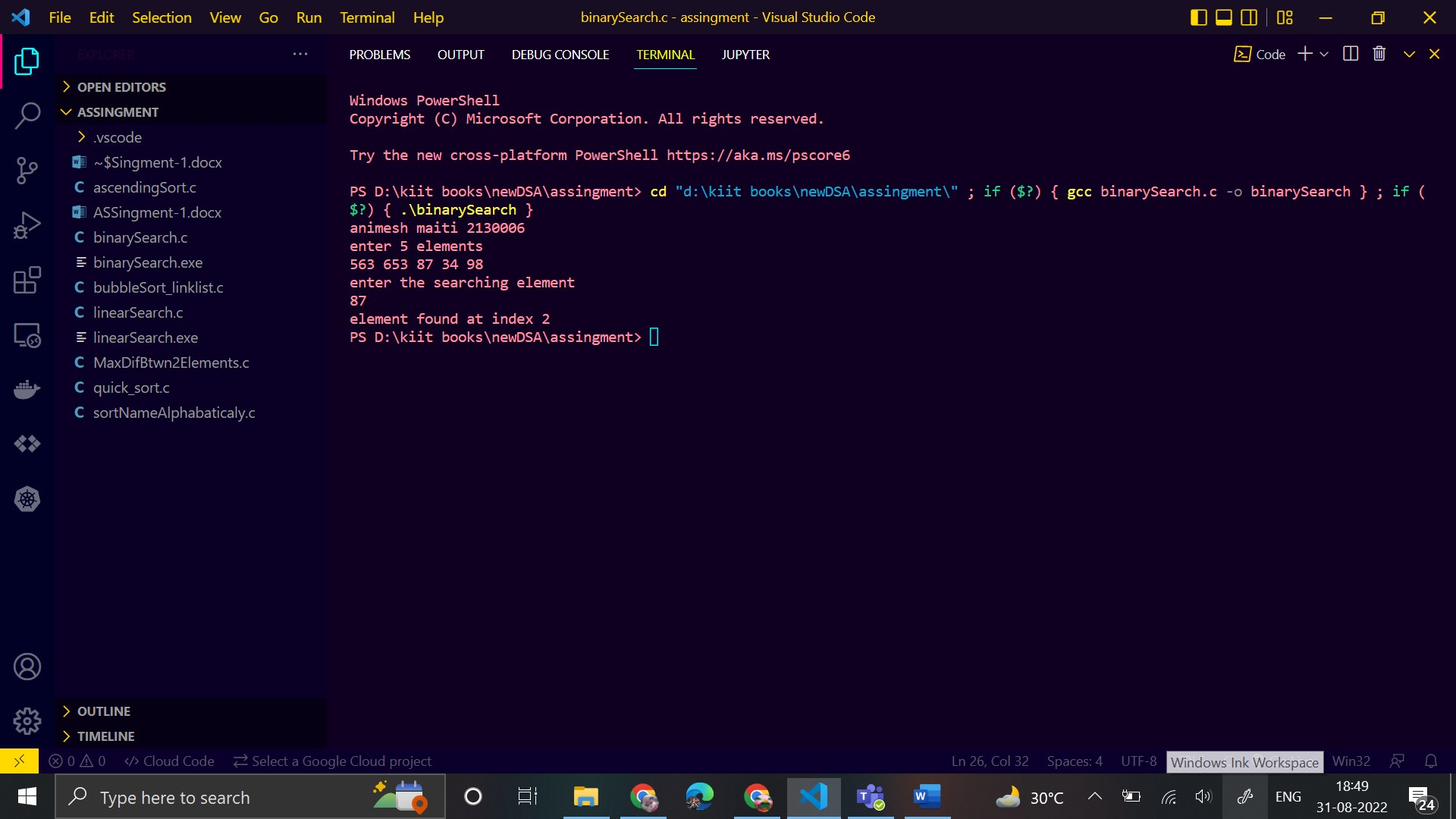
    {

        printf("element found at index %d\n",res);

    }

    return 0;

}



Q3. WAP to sort the given array, Arr = {82, 42, 49, 8, 25, 52, 36, 93, 59} in an ascending order. Use selection sort method.

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

    int temp = \*xp;

    \*xp = \*yp;

    \*yp = temp;

}

void selectionSort(int arr[], int n)

{

    int i, j, minIndex;

*// One by one move boundary of unsorted subarray*

    for (i = 0; i < n - 1; i++)

    {

*// Find the minimum element in unsorted array*

        minIndex = i;

        for (j = i + 1; j < n; j++)

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

                minIndex = j;

*// Swap the found minimum element with the first element*

        if (minIndex != i)

            swap(&arr[minIndex], &arr[i]);

        }

}

void printArray(int arr[], int size)

{

    for (int i = 0; i < size; i++)

        printf("%d ", arr[i]);

    printf("\n");

}

int main()

{

    printf("Animesh maiti\n");

    int arr[] = {82, 42, 49, 8, 25, 52, 36, 93, 59};

    int n = sizeof(arr) / sizeof(arr[0]);

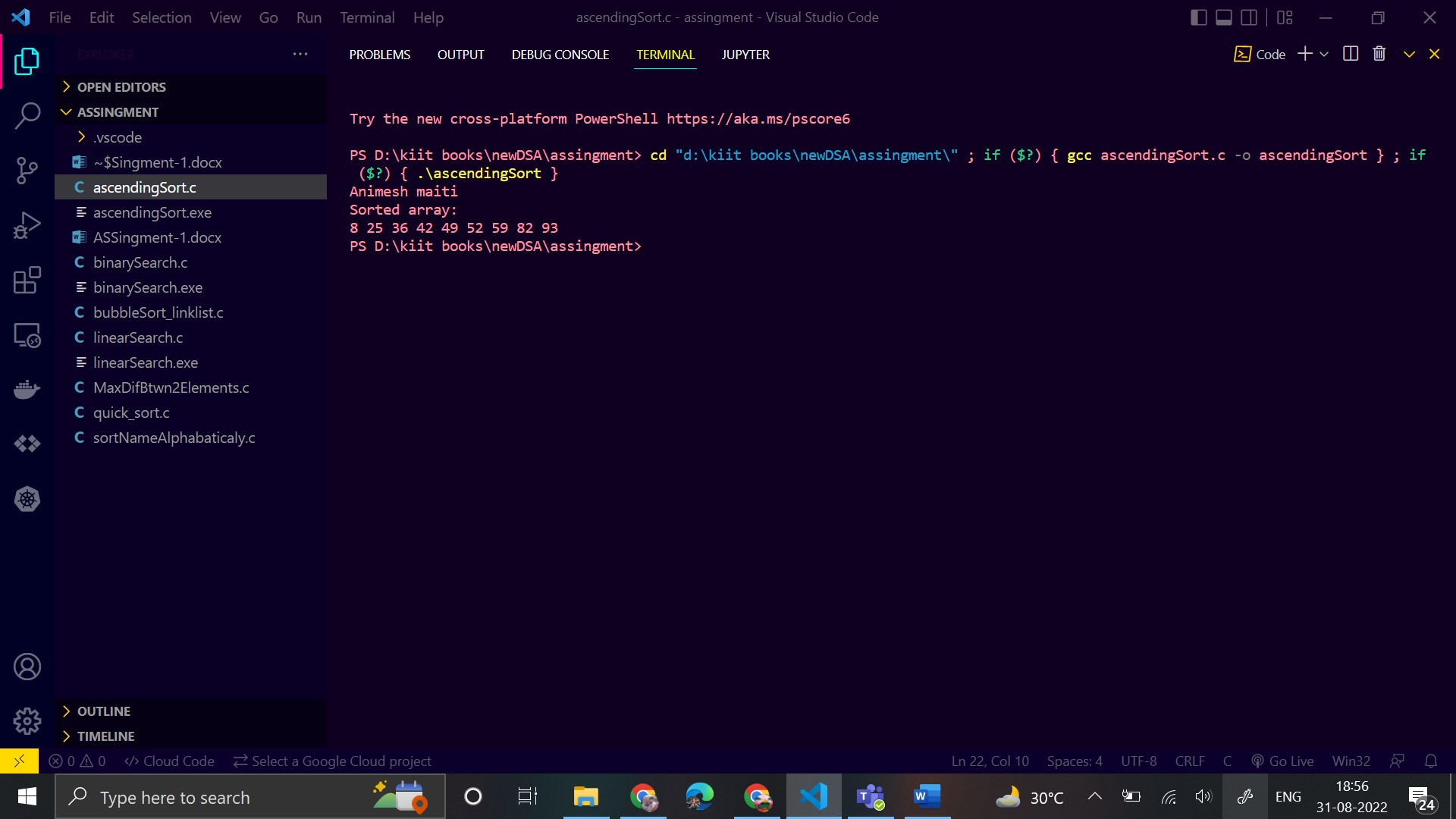
    selectionSort(arr, n);

    printf("Sorted array: \n");

    printArray(arr, n);

    return 0;

}



Q4. *WAP to sort an array of n doubles in a descending order using quick sort.*

*// WAP to sort an array of n doubles in a descending order using quick sort.*

#include <stdio.h>

*// A function to swap two elements*

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

{

    int t = \*a;

    \*a = \*b;

    \*b = t;

}

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

{

    int last = arr[high];

    int i = (low - 1);

    for (int j = low; j <= high - 1; j++)

    {

*// If current element is smaller than the last*

        if (arr[j] > last)

        {

            i++; *// increment index of smaller element*

            swap(&arr[i], &arr[j]);

        }

    }

    swap(&arr[i + 1], &arr[high]);

    return (i + 1);

}

void quickSort(int arr[], int low, int high)

{

    if (low < high)

    {

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

*// Separately sort elements before*

*// partition and after partition*

        quickSort(arr, low, pi - 1);

        quickSort(arr, pi + 1, high);

    }

}

*// Function to print an array*

void printArray(int arr[], int size)

{

    int i;

    for (i = 0; i < size; i++)

        printf("%d ", arr[i]);

}

int main()

{

    printf("animesh maiti\n");

    int arr[] = {10, 7, 8, 9, 1, 5};

    int n = sizeof(arr) / sizeof(arr[0]);

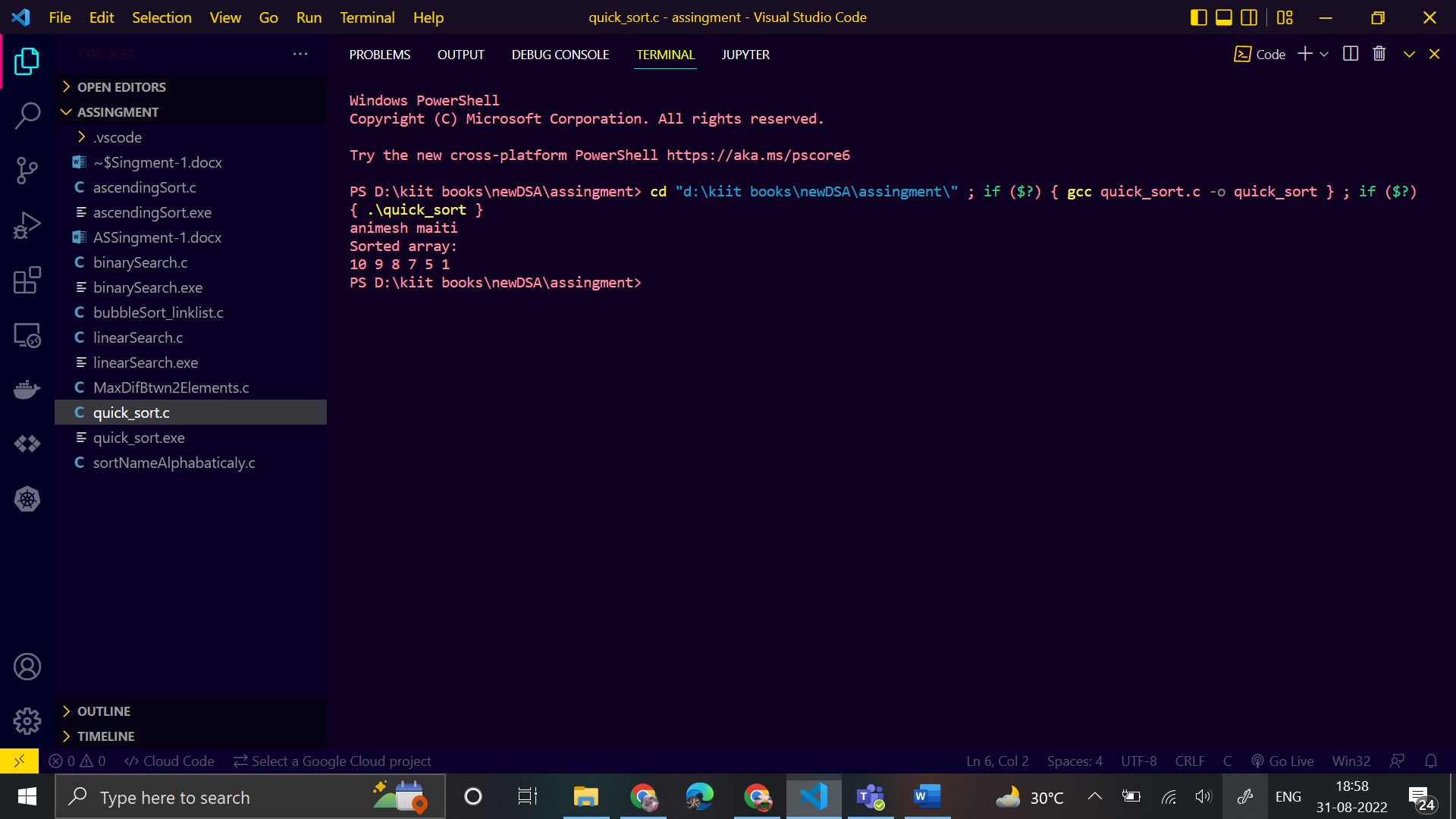
    quickSort(arr, 0, n - 1);

    printf("Sorted array: \n");

    printArray(arr, n);

    return 0;

}



Q5. *WAP sort the n names in an alphabetical order.*

*// WAP sort the n names in an alphabetical order.*

#include <stdio.h>

#include <string.h>

int main()

{

    printf("animesh maiti\n");

    int i, j, n;

    char str[100][100], s[100];

    printf("Enter number of names :\n");

    scanf("%d", &n);

    printf("Enter names in any order:\n");

    for (i = 0; i < n; i++)

    {

        scanf("%s", str[i]);

    }

    for (i = 0; i < n; i++)

    {

        for (j = i + 1; j < n; j++)

        {

            if (strcmp(str[i], str[j]) > 0)*//if ascii value is greater than 2nd strcmp give value 1 than swap if equal return 0 if less return -1*

            {

*// swaping names by copying*

                strcpy(s, str[i]);

                strcpy(str[i], str[j]);

                strcpy(str[j], s);

            }

        }

    }

    printf("\nThe sorted order of names are:\n");

    for (i = 0; i < n; i++)

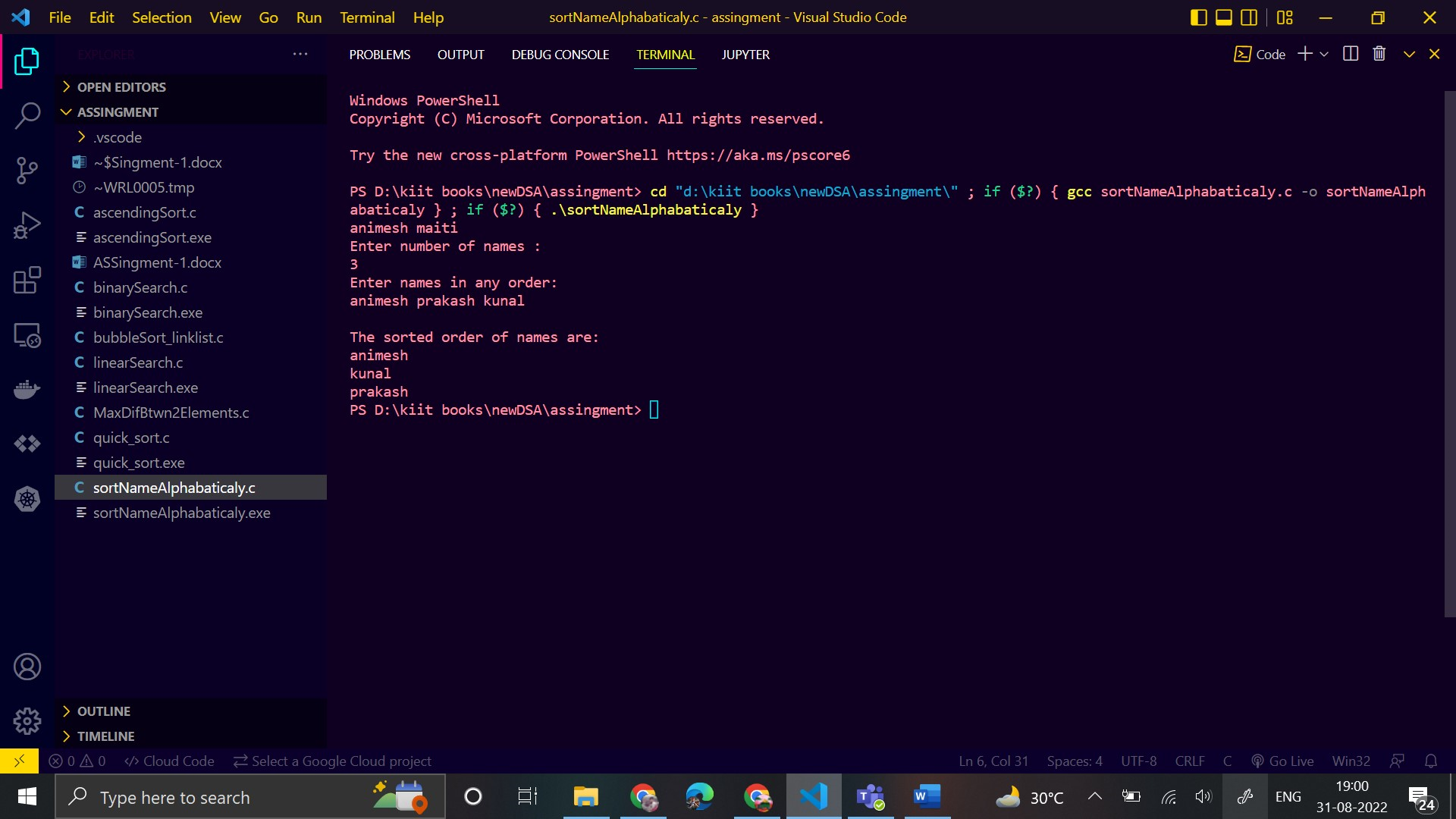
    {

        printf("%s\n", str[i]);

    }

    return 0;

}



Q6. WAP demonstrating bubble sort using linked list.

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

int main()

{

    struct node \*temp1, \*temp2, \*ptr, \*newNode, \*startList;

    int n, k, i, j;

    startList = NULL;

    printf("Animesh Maiti\n");

    printf("Input number of elements\n");

    scanf("%d", &n);

    printf("Input the elements:\n");

    for (i = 1; i <= n; i++)

    {

        if (startList == NULL)

        {

            newNode = (struct node \*)malloc(sizeof(struct node));

            scanf("%d", &newNode->data);

            newNode->next = NULL;

            startList = newNode;

            temp1 = startList;

        }

        else

        {

            newNode = (struct node \*)malloc(sizeof(struct node));

            scanf("%d", &newNode->data);

            newNode->next = NULL;

            temp1->next = newNode;

            temp1 = newNode;

        }

    }

    for (i = n - 2; i >= 0; i--)

    {

        temp1 = startList;

        temp2 = temp1->next;

        for (j = 0; j <= i; j++)

        {

            if (temp1->data > temp2->data)

            {

                k = temp1->data;

                temp1->data = temp2->data;

                temp2->data = k;

            }

            temp1 = temp2;

            temp2 = temp2->next;

        }

    }

    printf("Sorted order is: \n");

    ptr = startList;

    while (ptr != NULL)

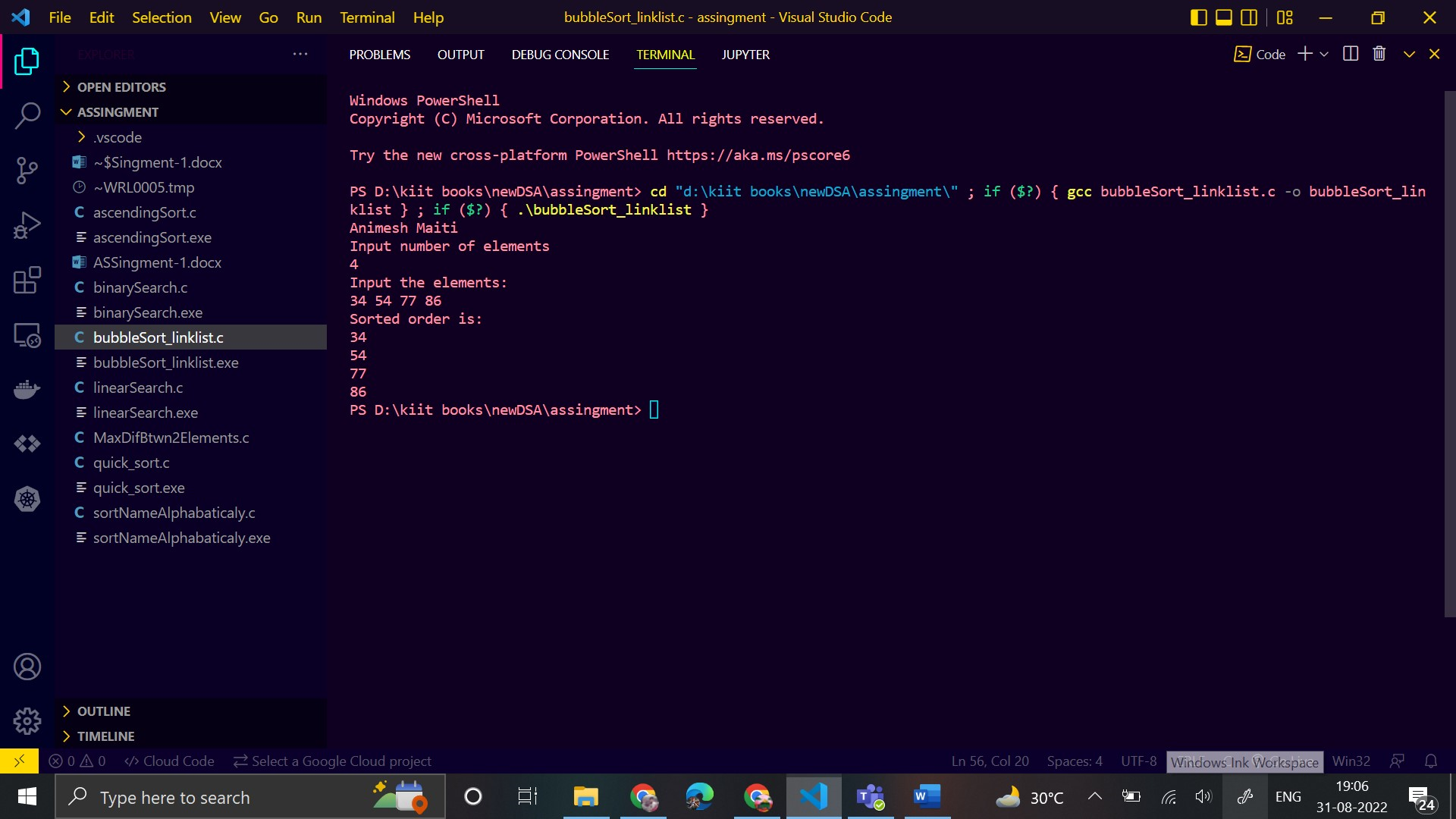
    {

        printf("%d\n", ptr->data);

        ptr = ptr->next;

    }

}



Q7. WAP to find the maximum difference between any two elements

#include <stdio.h>

int maxDiff(int arr[], int arr\_size)

{

    int maxDiff = arr[1] - arr[0];

    int min = arr[0];

    int i;

    for (i = 1; i < arr\_size; i++)

    {

        if (arr[i] - min > maxDiff)

            maxDiff = arr[i] - min;

        if (arr[i] < min)

            min = arr[i];

    }

    return maxDiff;

}

int main()

{

    printf("animesh maiti\n");

    int arr[] = {20, 13, 53, 84, 10};

    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Maximum difference is %d", maxDiff(arr, size));

    getchar();

    return 0;

}

