**Assignment: 10**

**EXP .** WAP to populate an array of 'n' elements using a random function. Share the time complexity for all the experiments, n should be large enough to see the difference in execution.

1. Implement Insertion sort in the above data set.
2. Implement Selection sort in the above data set.
3. Implement Merge Sort using the above dataset.
4. Implement Quicksort in the above data set.

**CODING:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

double time1, timedif;

void random(int array[], int length, int max)

{

    for (int i = 0; i < length; i++)

        array[i] = (rand() % max) + 1;

}

void printArray(int array[], int size\_arr)

{

    for (int i = 0; i < size\_arr; ++i)

    {

        printf("%d\n ", array[i]);

    }

    printf("\n");

}

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

{

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

}

void selectionSort(int array[], int size\_arr)

{

    for (int step = 0; step < size\_arr - 1; step++)

    {

        int min\_idx = step;

        for (int i = step + 1; i < size\_arr; i++)

        {

            if (array[i] < array[min\_idx])

                min\_idx = i;

        }

        swap(&array[min\_idx], &array[step]);

    }

}

void insertionSort(int array[], int size\_arr)

{

    for (int step = 1; step < size\_arr; step++)

    {

        int key = array[step];

        int j = step - 1;

        while (key < array[j] && j >= 0)

        {

            array[j + 1] = array[j];

            --j;

        }

        array[j + 1] = key;

    }

}

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

{

    int pivot = array[high];

    int i = (low - 1);

    for (int j = low; j < high; j++)

    {

        if (array[j] <= pivot)

        {

            i++;

            swap(&array[i], &array[j]);

        }

    }

    swap(&array[i + 1], &array[high]);

    return (i + 1);

}

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

{

    if (low < high)

    {

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

        quickSort(array, low, pi - 1);

        quickSort(array, pi + 1, high);

    }

}

int main(void)

{

    srand(time(NULL));

    int size\_arr;

    int choice;

    int array[size\_arr];

    printf("Please enter the number of random numbers to be genrated:\n");

    scanf("%d", &size\_arr);

    random(array, size\_arr, 1000000000);

    for (int i = 0; i < size\_arr; i++)

    {

        printf("a[%d]=%d\n", i, array[i]);

    }

    printf("\n\t 1.selection sort\n\t 2.insertion sort\n\t 3.quick sort\n\t 4.EXIT");

    do

    {

        printf("\n Enter the Choice:");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

        {

            time1 = (double)clock();

            time1 = time1 / CLOCKS\_PER\_SEC;

            selectionSort(array, size\_arr);

            printf("Sorted array for selection sort:\n");

            printArray(array, size\_arr);

            timedif = (((double)clock()) / CLOCKS\_PER\_SEC) - time1;

            printf("The elapsed time is %f seconds\n", timedif);

            break;

        }

        case 2:

        {

            time1 = (double)clock();

            time1 = time1 / CLOCKS\_PER\_SEC;

            insertionSort(array, size\_arr);

            printf("Sorted arrayfor insertion sort:\n");

            printArray(array, size\_arr);

            timedif = (((double)clock()) / CLOCKS\_PER\_SEC) - time1;

            printf("The elapsed time is %f seconds\n", timedif);

            break;

        }

        case 3:

        {

            time1 = (double)clock();

            time1 = time1 / CLOCKS\_PER\_SEC;

            quickSort(array, 0, size\_arr - 1);

            printf("Sorted array for quick sort: \n");

            printArray(array, size\_arr);

            timedif = (((double)clock()) / CLOCKS\_PER\_SEC) - time1;

            printf("The elapsed time is %f seconds\n", timedif);

            break;

        }

        case 4:

        {

            printf("\n\t EXIT POINT ");

            break;

        }

        default:

        {

            printf("\n\t Please enter a Valid Choice(1/2/3/4)");

        }

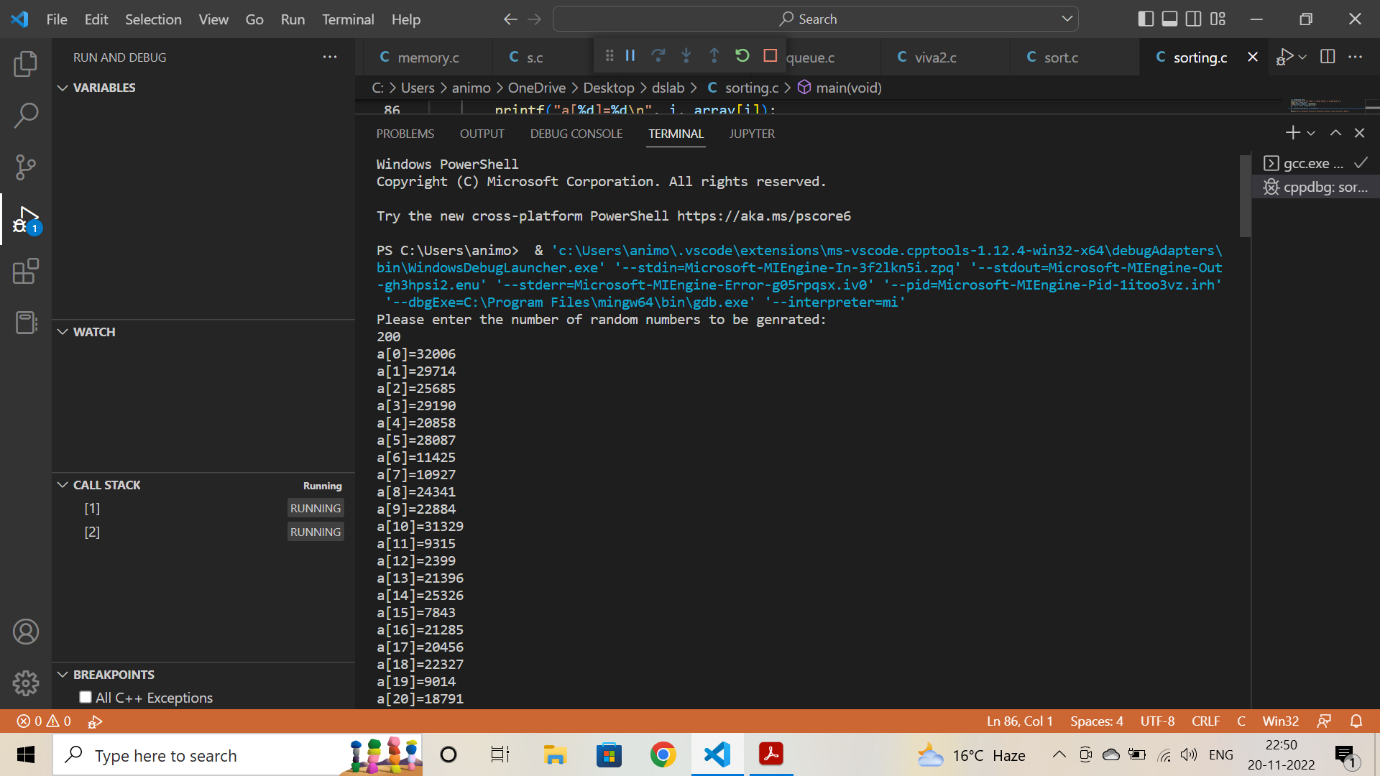
        }

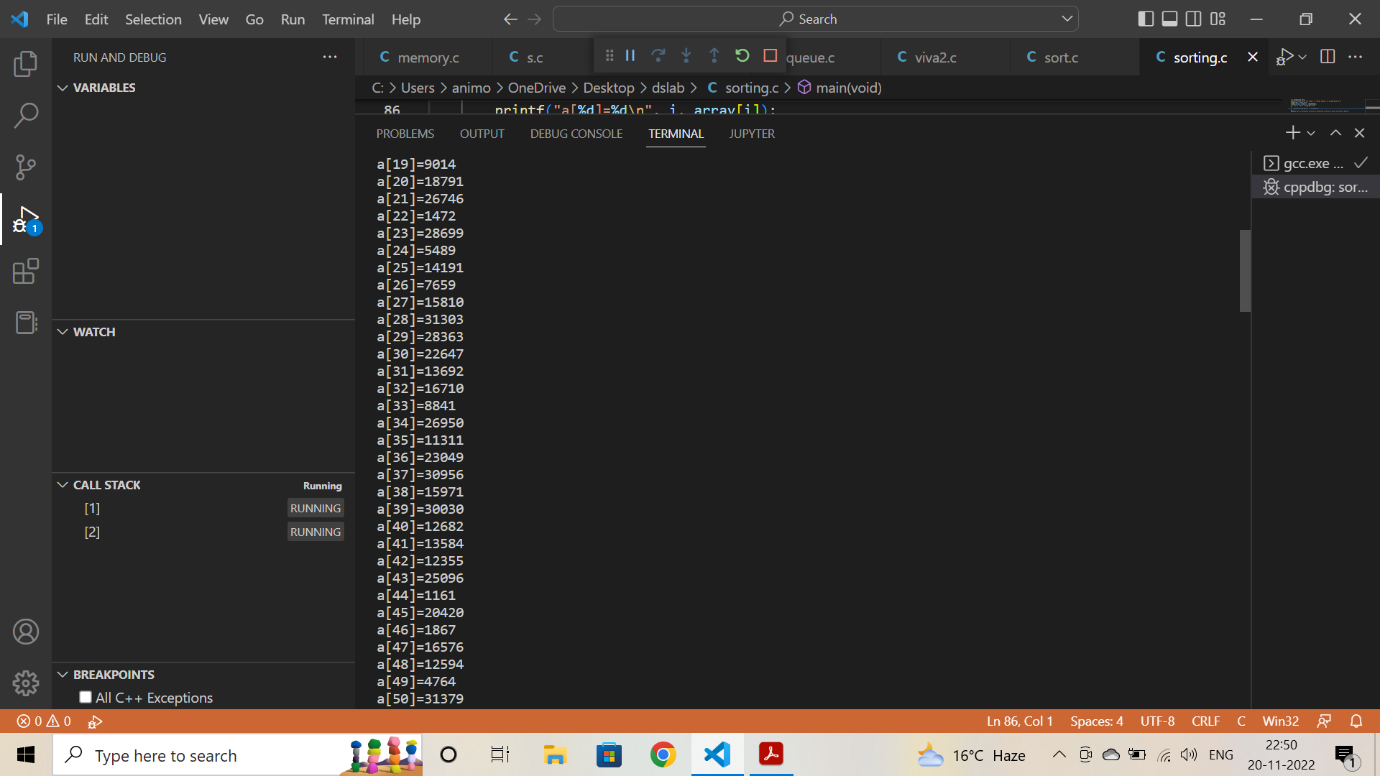
    } while (choice != 4);

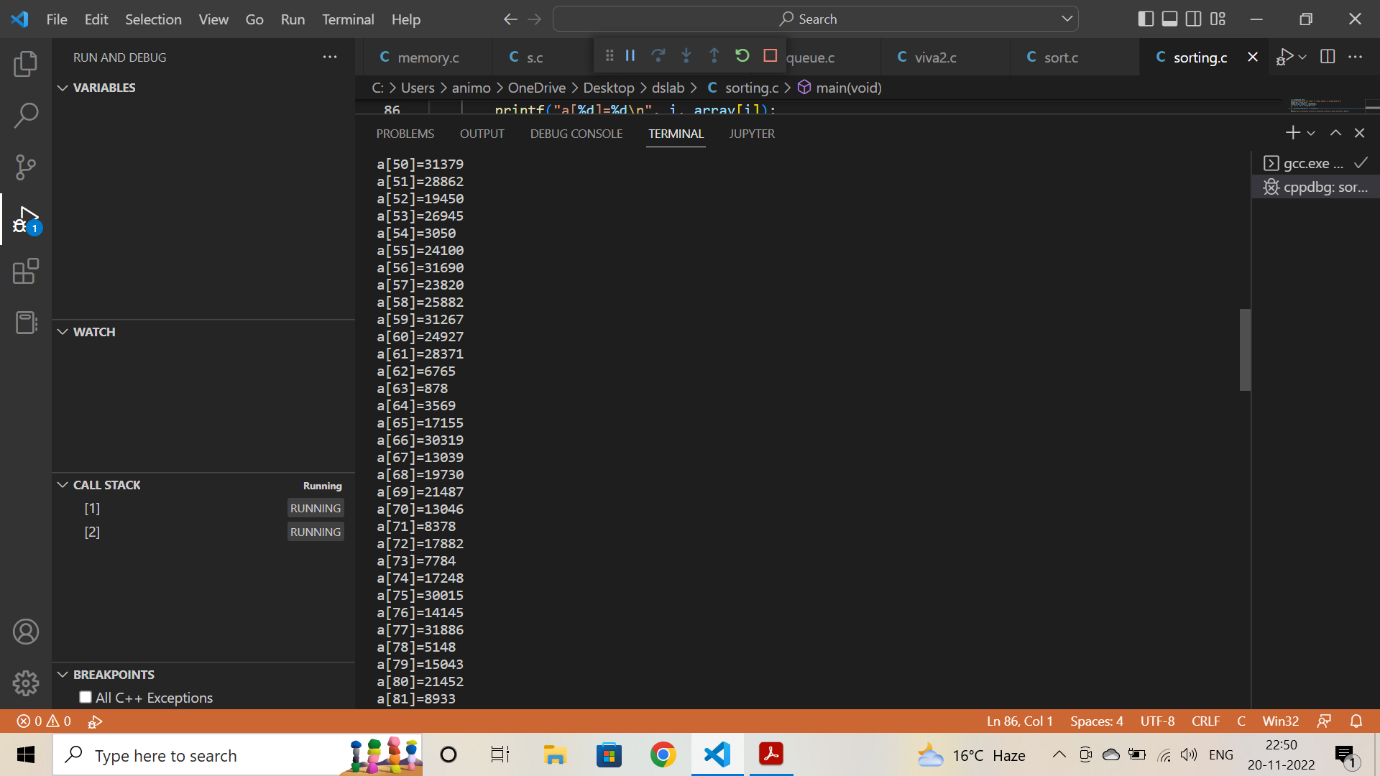
    return 0;

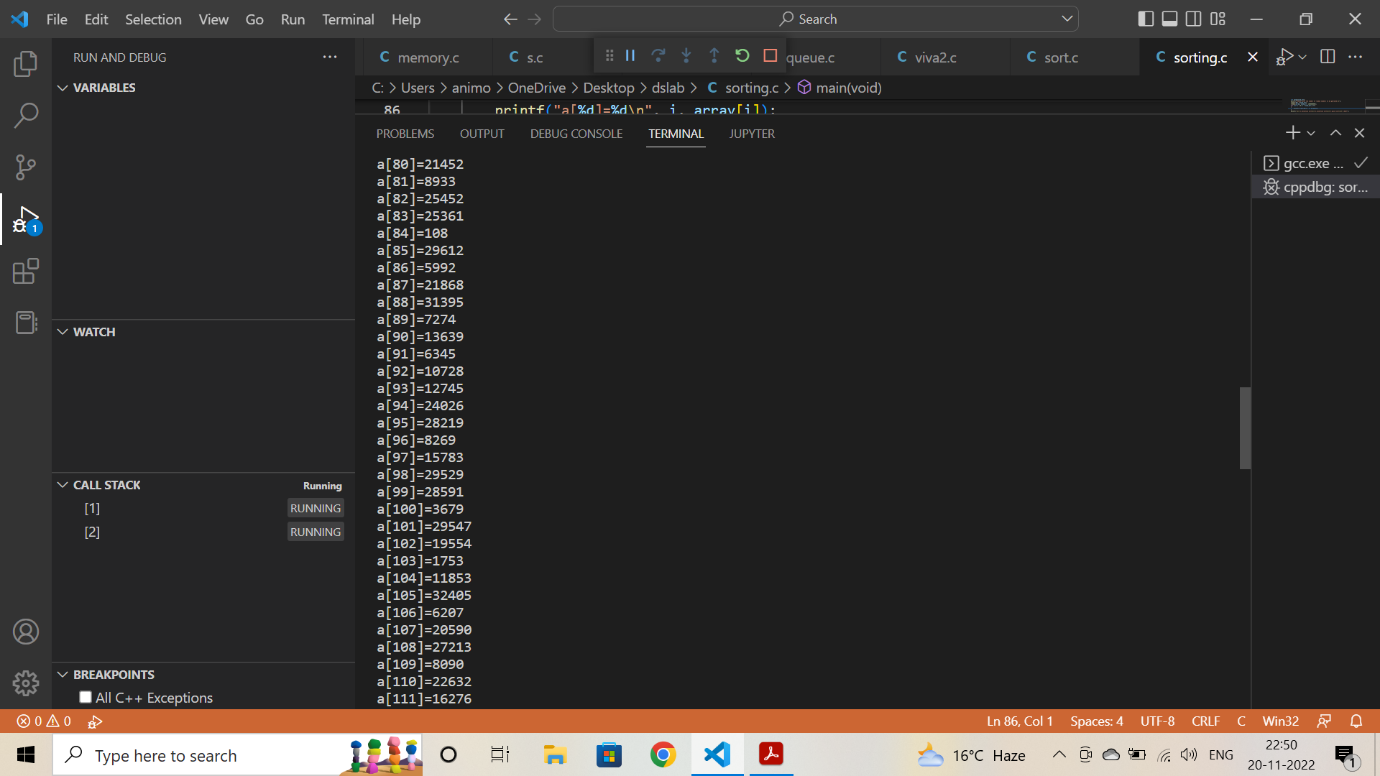
}

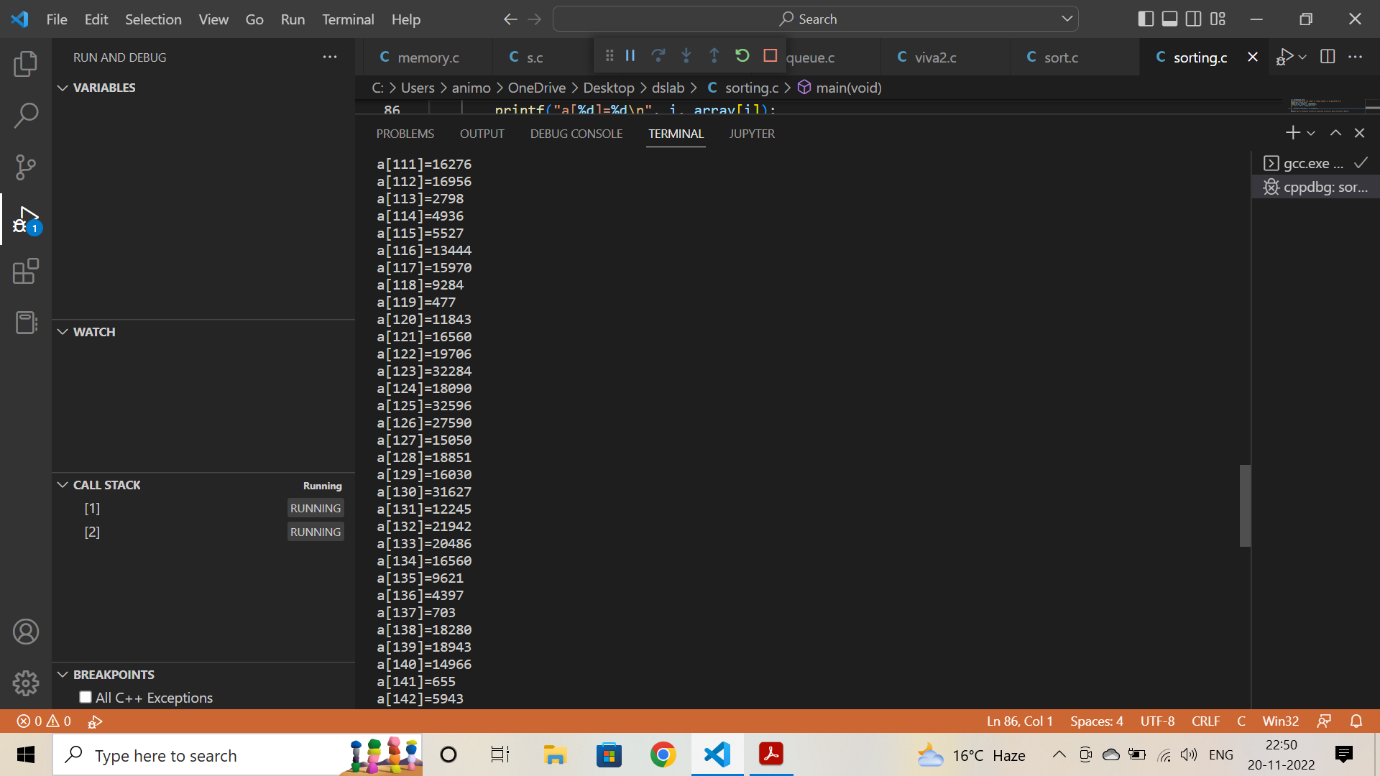
**OUTPUT:** 200 random number array.

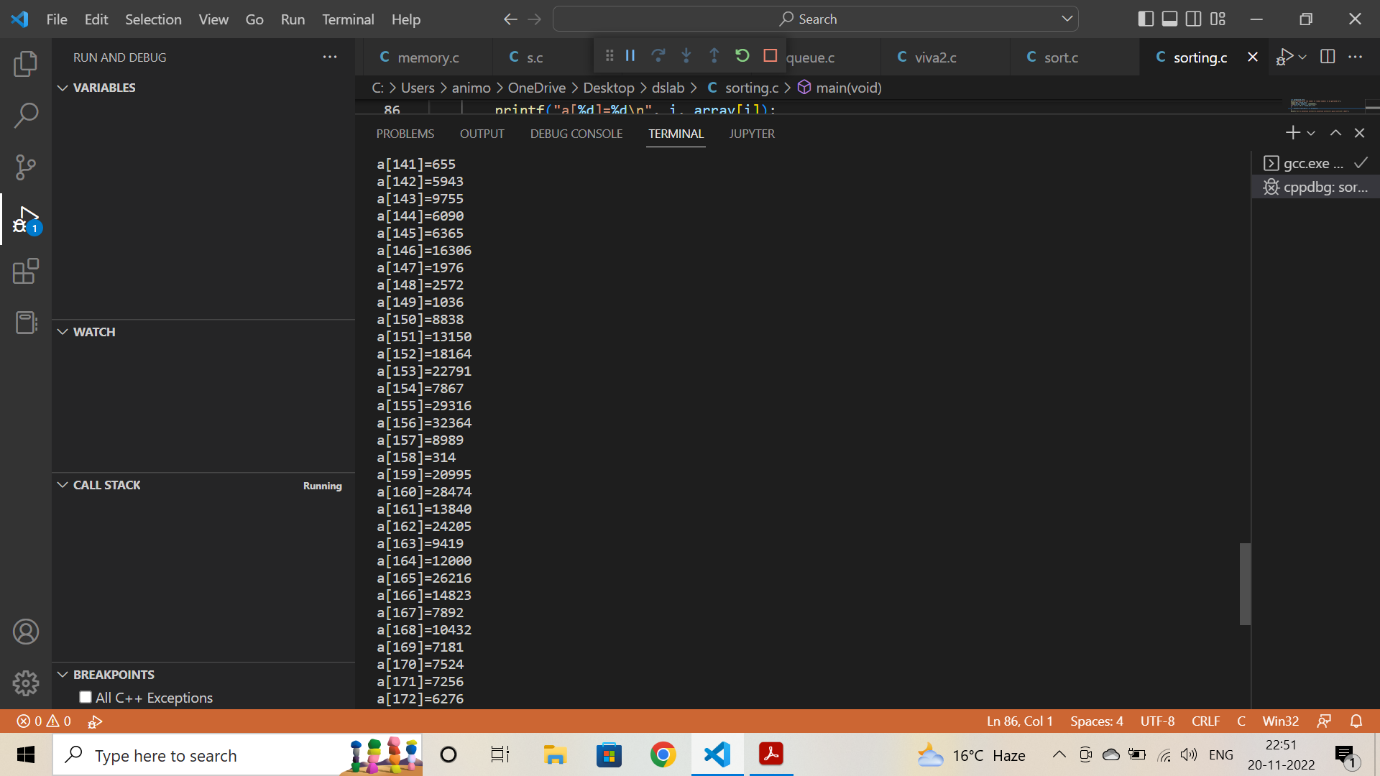


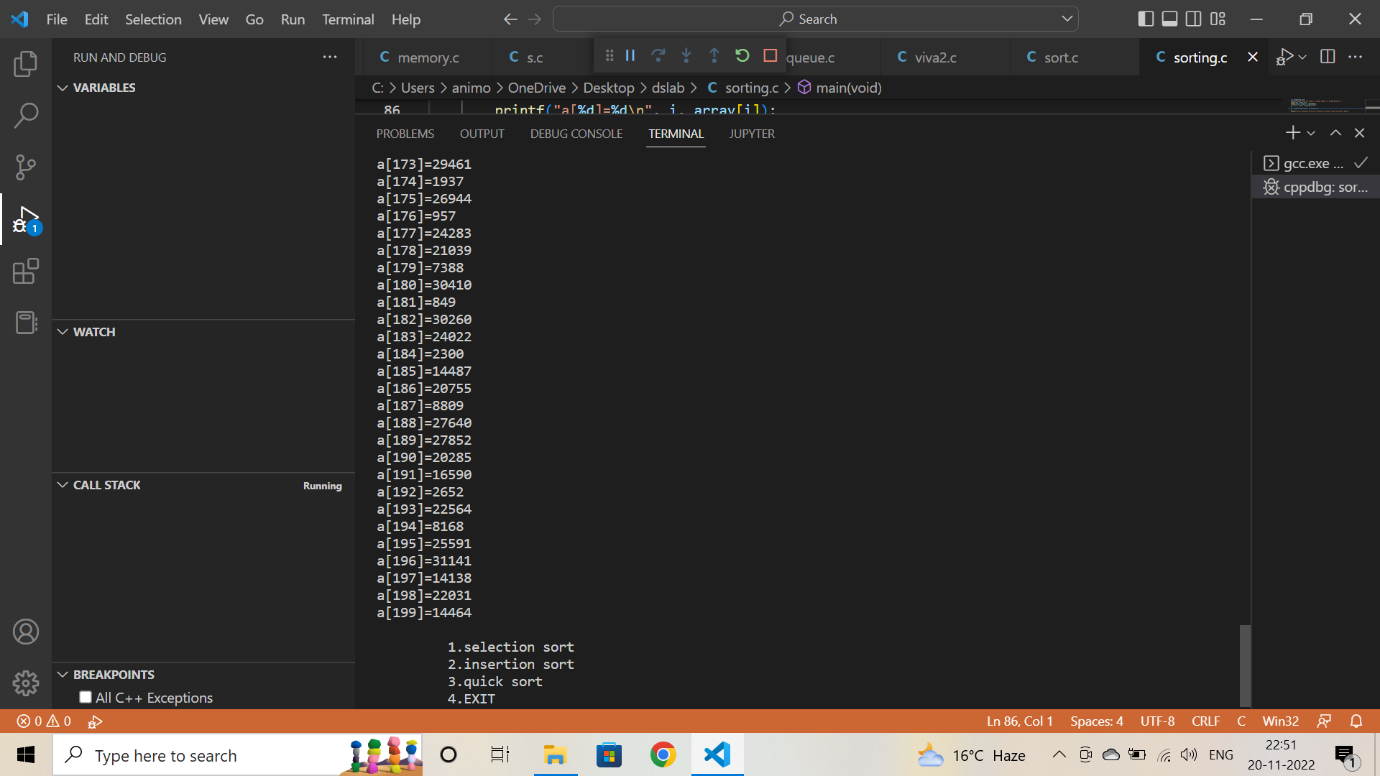




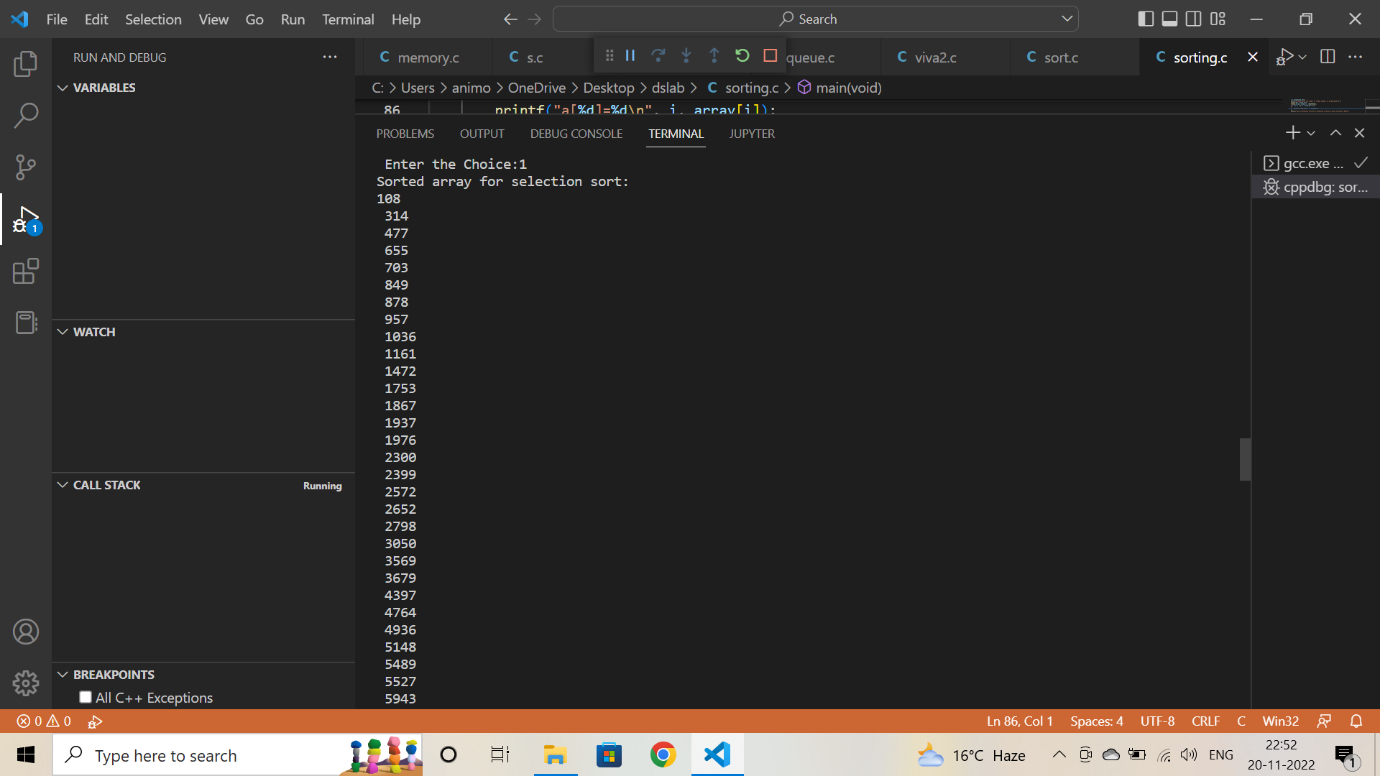


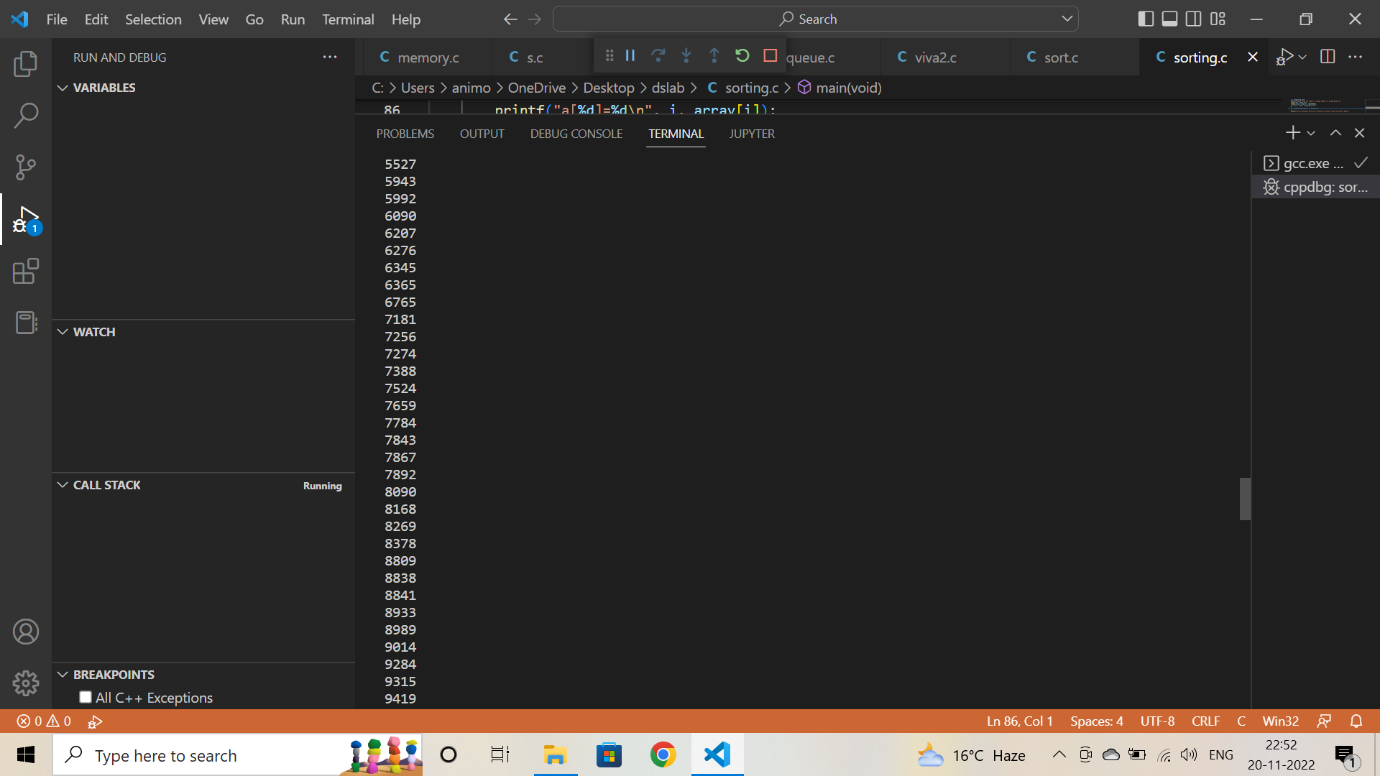
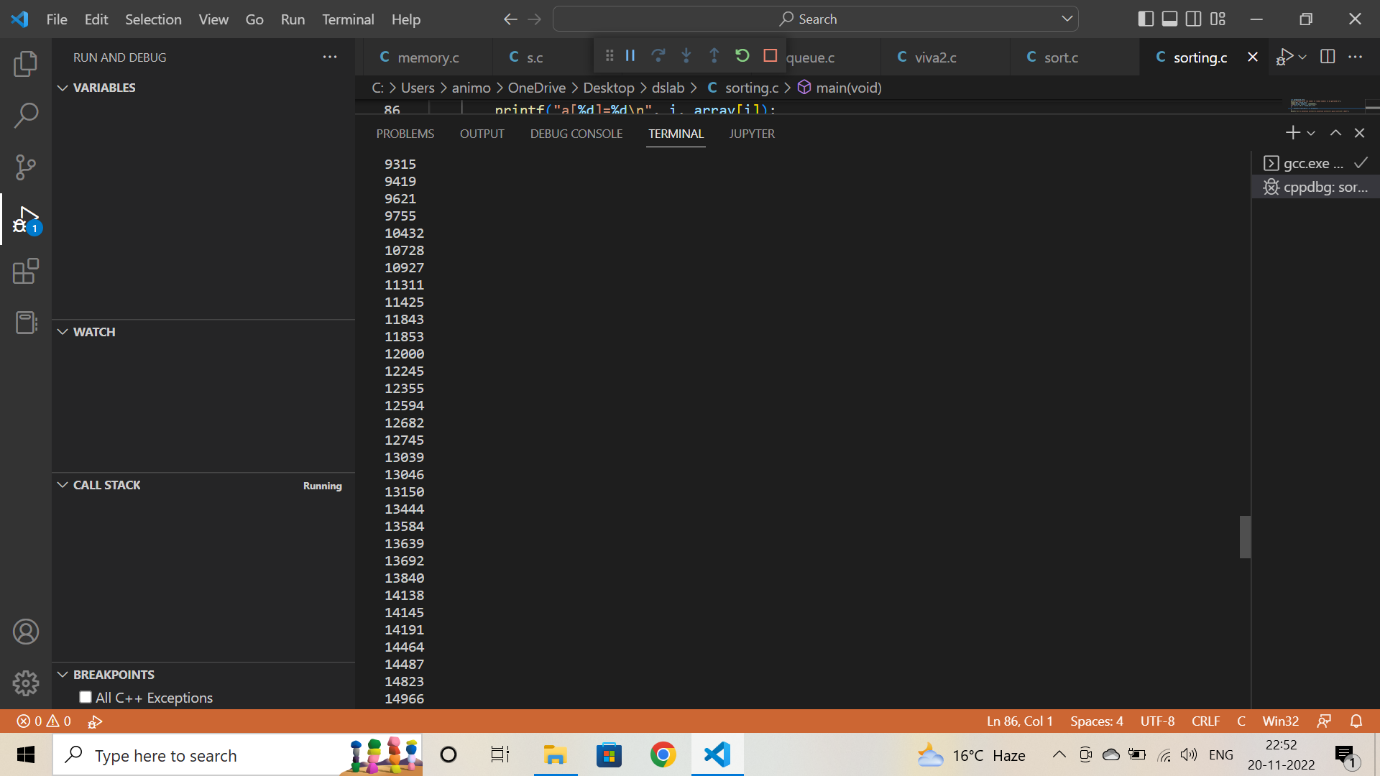


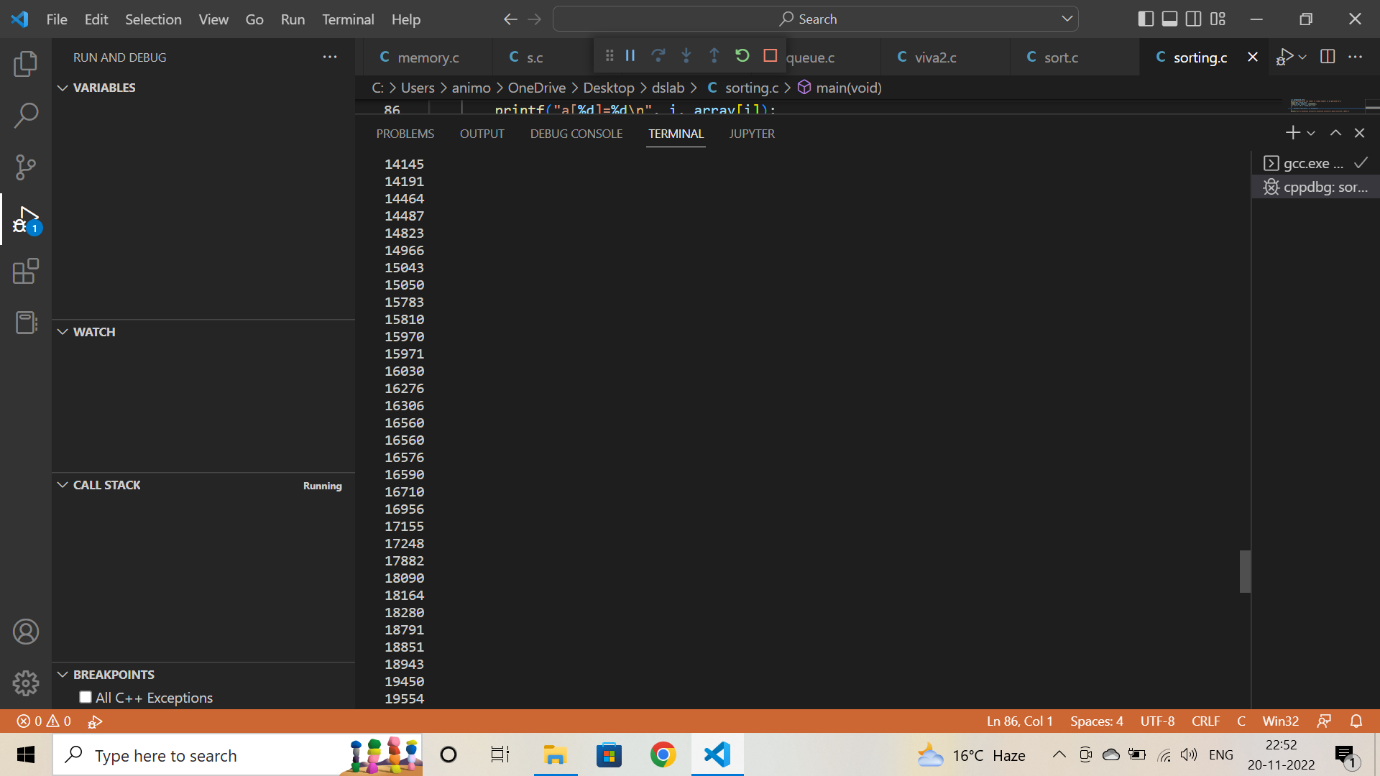


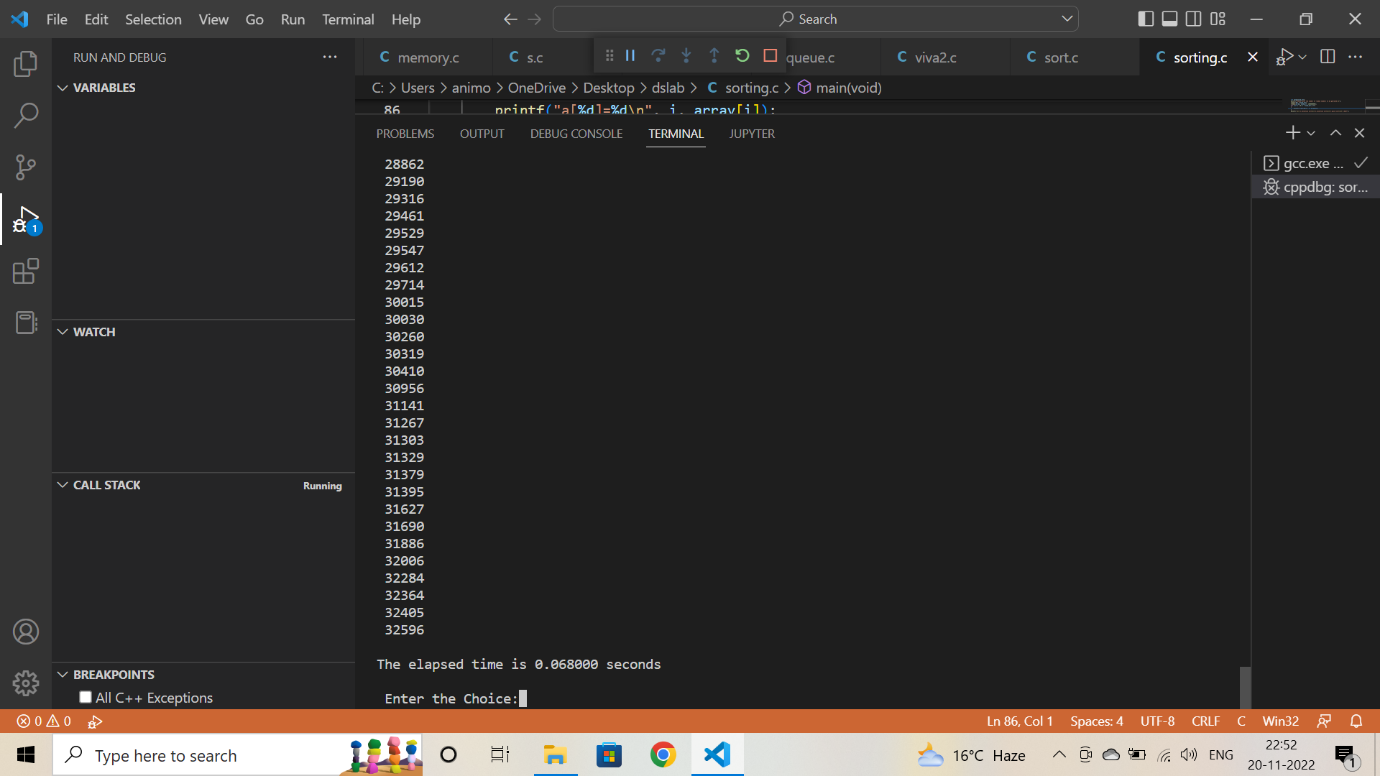


1. **Selection sort**



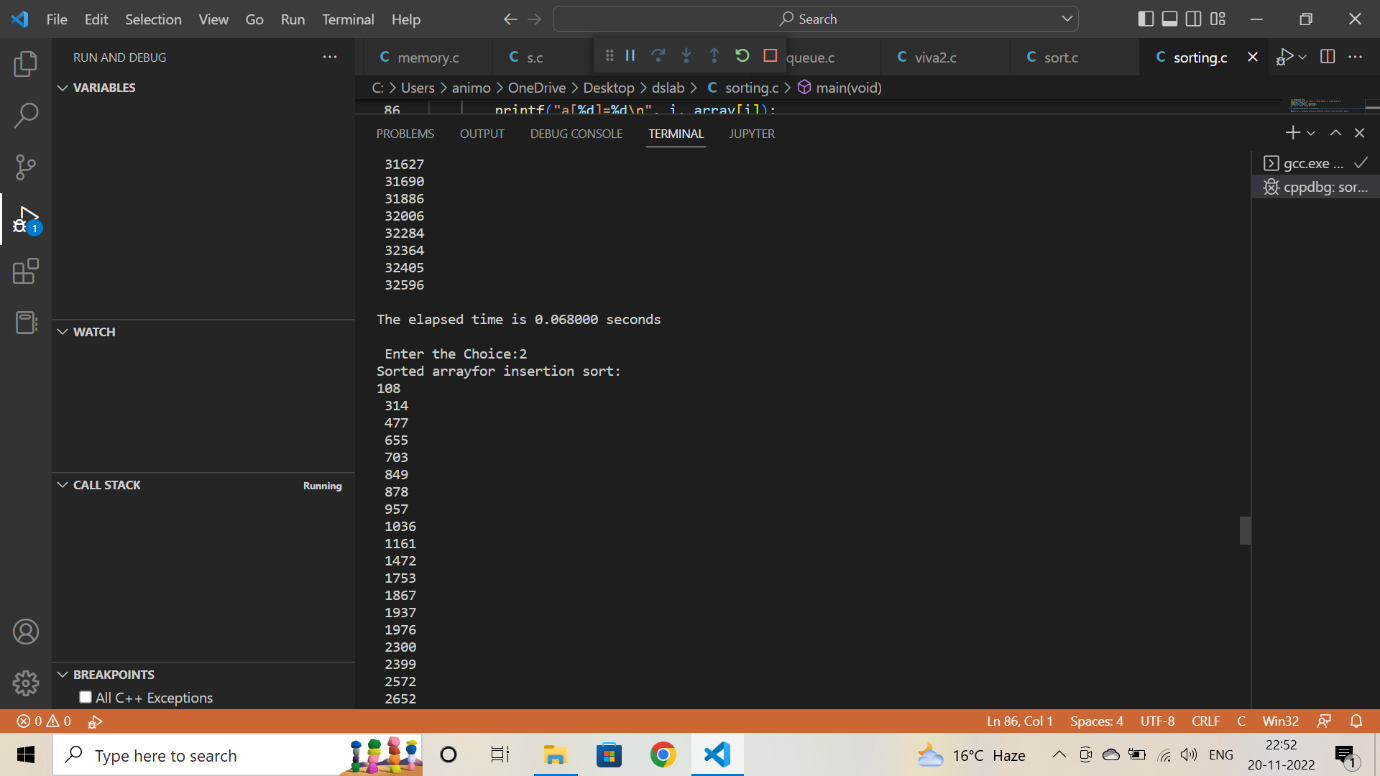
 

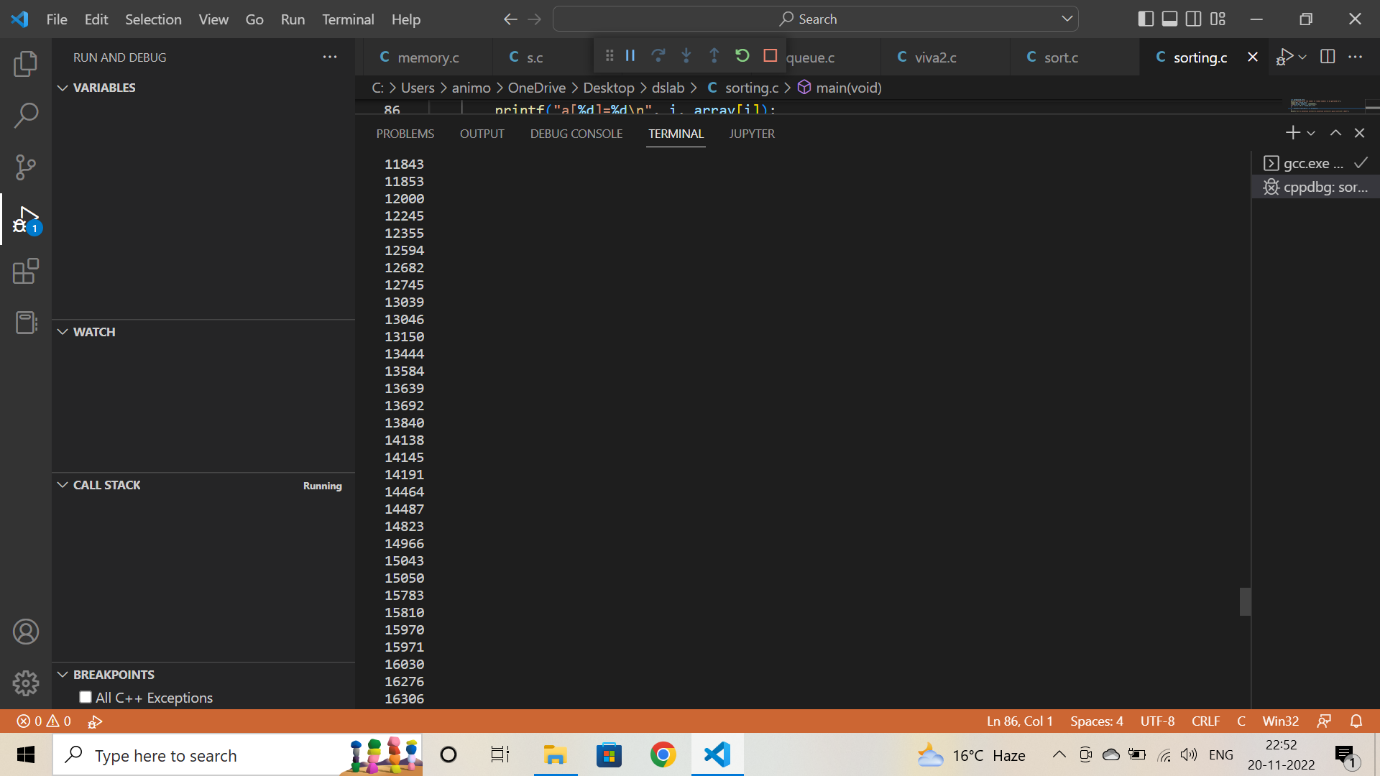
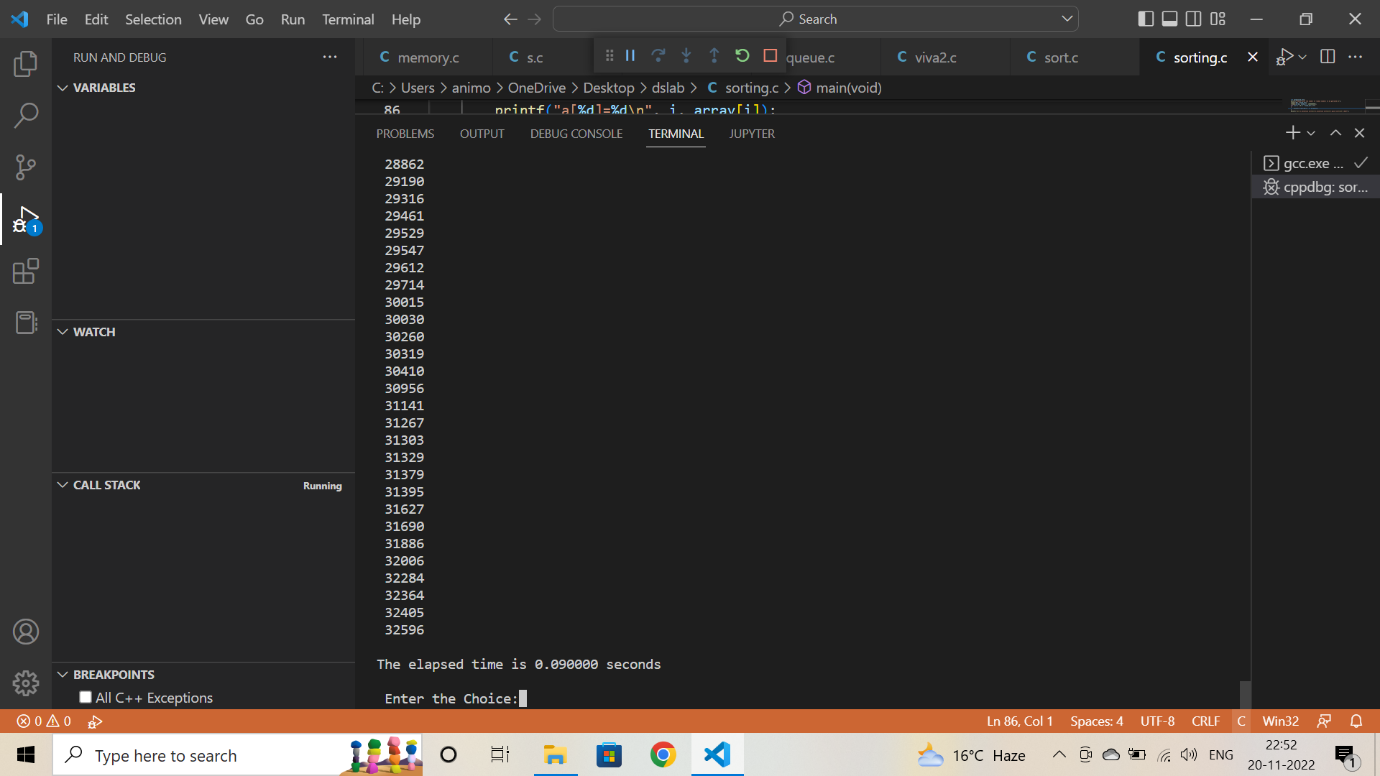




**Time elapsed 0.068000 second.**

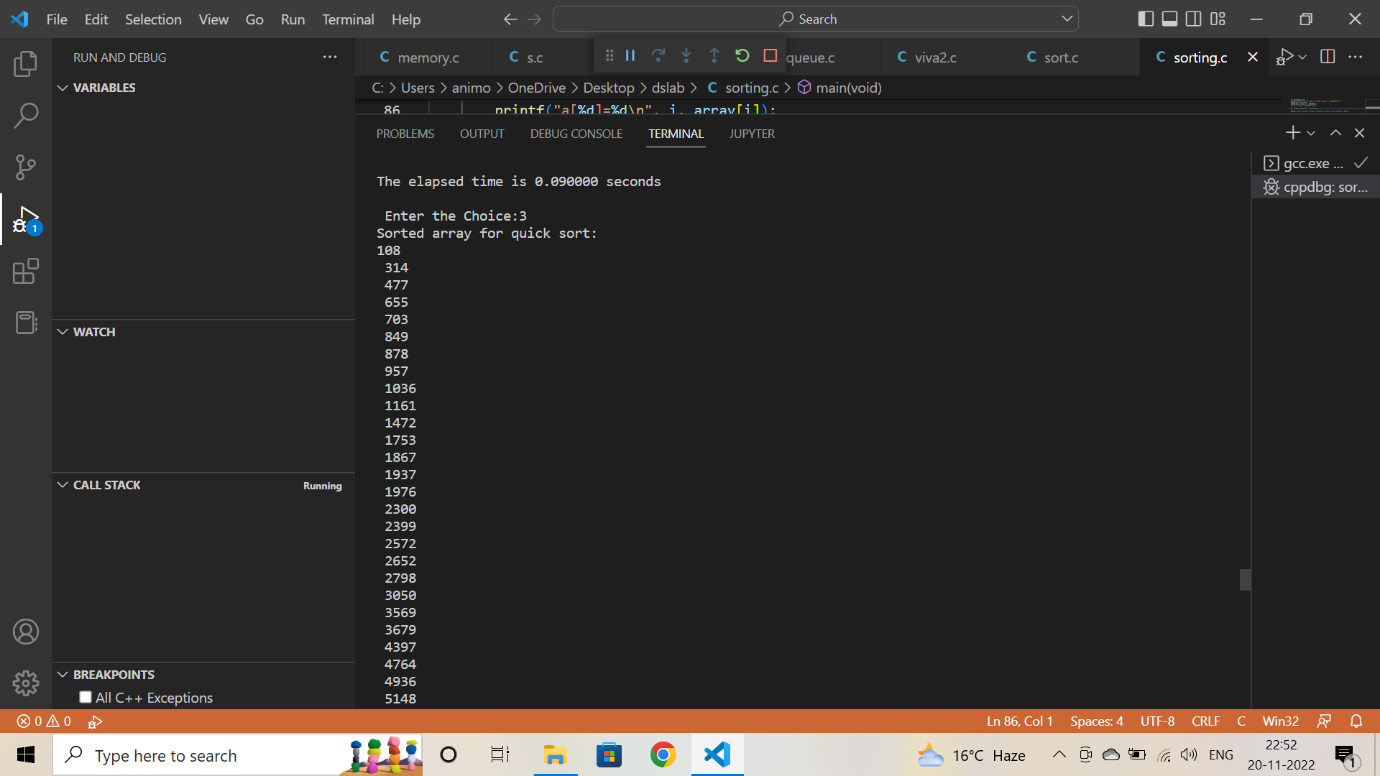
1. **Insertion sort**

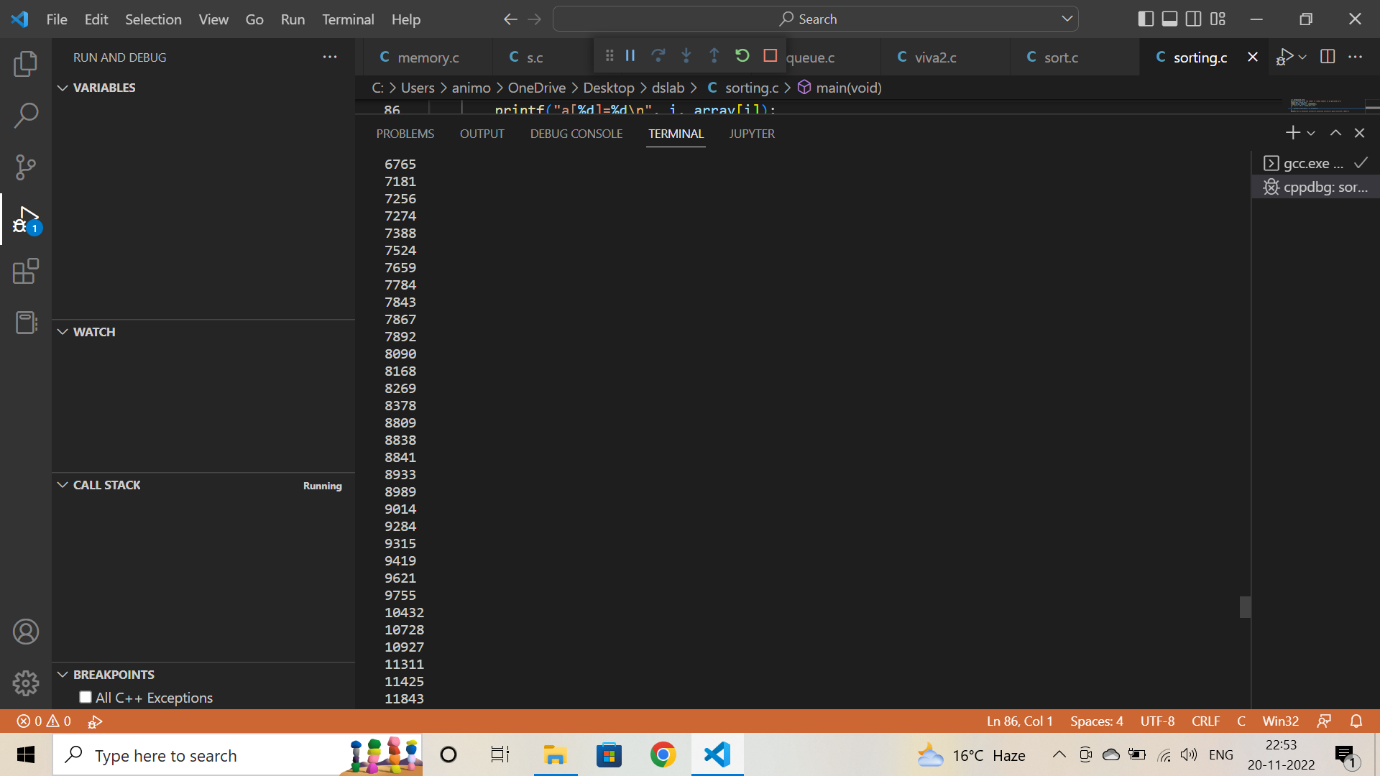


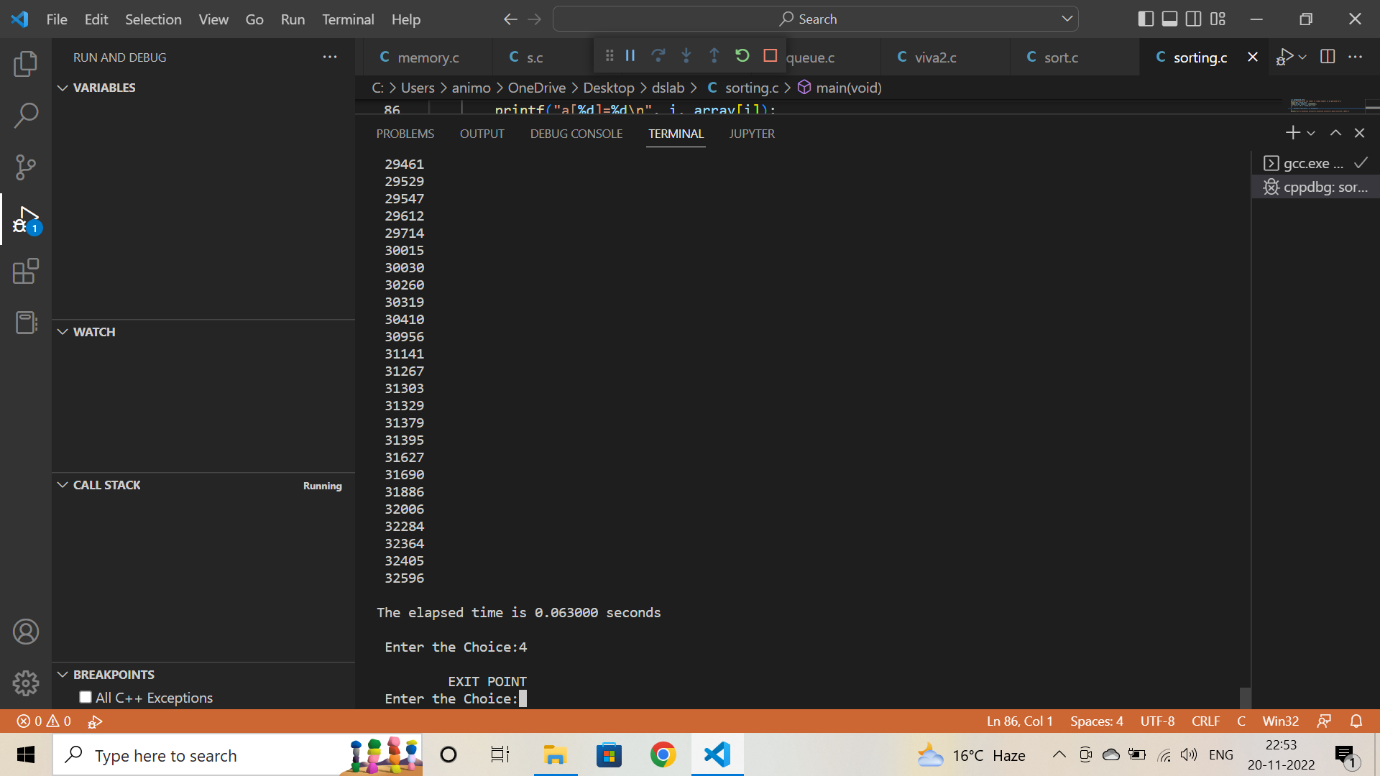
 

**Time elapsed 0.090000 second.**

1. **Quicksort**







**Time elapsed 0.063000 second.**