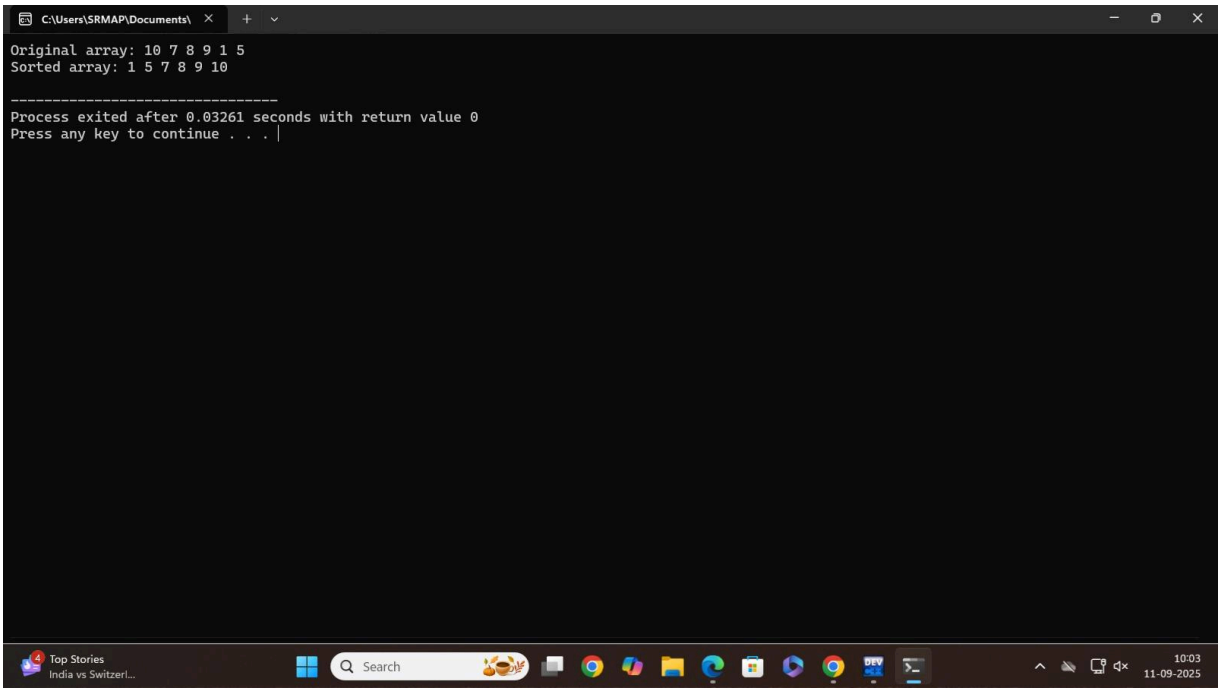
1.Quick sort

```
#include <stdio.h>
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            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);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    printArray(arr, n);

    quickSort(arr, 0, n - 1);

    printf("Sorted array: ");
```

```
    printArray(arr, n);

    return 0;
}
```



2.Strassens algorithm for matrix multiplication

```
#include <stdio.h>

void strassen(int A[2][2], int B[2][2], int C[2][2]) {
    int M1 = (A[0][0] + A[1][1]) * (B[0][0] + B[1][1]);
    int M2 = (A[1][0] + A[1][1]) * B[0][0];
    int M3 = A[0][0] * (B[0][1] - B[1][1]);
    int M4 = A[1][1] * (B[1][0] - B[0][0]);
    int M5 = (A[0][0] + A[0][1]) * B[1][1];
    int M6 = (A[1][0] - A[0][0]) * (B[0][0] + B[0][1]);
    int M7 = (A[0][1] - A[1][1]) * (B[1][0] + B[1][1]);

    C[0][0] = M1 + M4 - M5 + M7;
    C[0][1] = M3 + M5;
    C[1][0] = M2 + M4;
    C[1][1] = M1 - M2 + M3 + M6;
}

int main() {
```

```
    int A[2][2] = {{1, 3}, {7, 5}};
    int B[2][2] = {{6, 8}, {4, 2}};
    int C[2][2];

    strassen(A, B, C);

    printf("Resultant Matrix:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++)
            printf("%d ", C[i][j]);
        printf("\n");
    }
    return 0;
}
```