## Bangladesh Army International University of Science & Technology

# Department of Computer Science and Engineering

## Lab Report

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Lab Report No** | **04** | | | | | | |
| **Lab Report Name** | **Sorting an Array Using Quick Sort** | | | | | | |
| **Course Title** | **Computer Algorithms & Complexity Sessional** | | | | | | |
| **Course Code** | **CSE 222** | | | | | | |
| **Name** | Rayhan Alam Towhid | | | | | | |
| **ID** | 1119007 | | | | | | |
| **Level** | 2 | **Term** | II | **Section** | A | **Group** | G1 |
| **Date of Submission** | 19-10-2025 | | | **Session** | Fall-2025 | | |

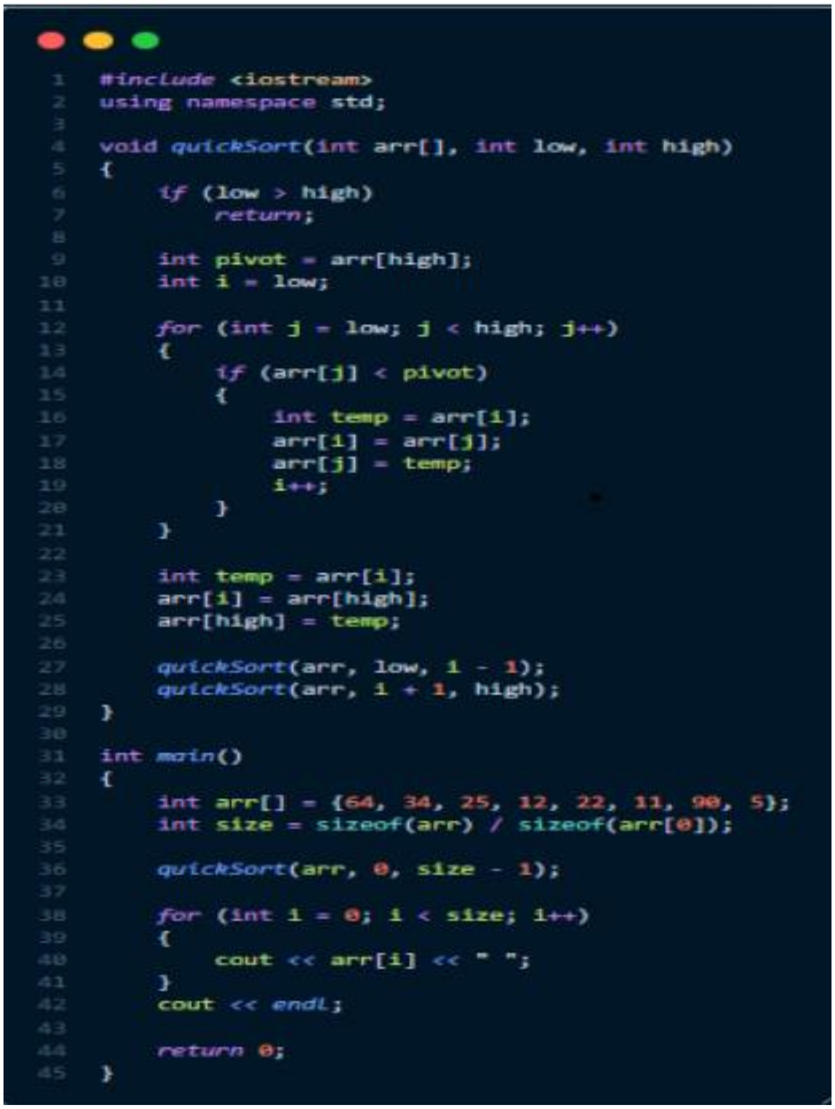
**Marking Rubric:**

| Problem Understanding & Report Clarity (3) | Implementation (5) | Results & Analysis (2) | Total (10) |
|---|---|---|---|
| | | | |

## Key Learnings:

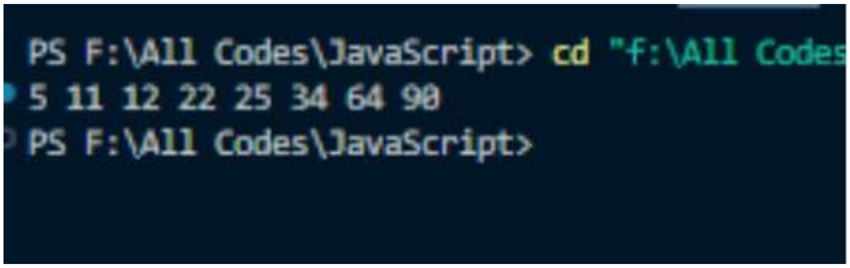
Quick Sort teaches how to divide and conquer by choosing a pivot, partitioning the array around it, and recursively sorting the subarrays. You also learn about in-place sorting and improving efficiency with recursion.

## Code Implementation:



```cpp
#include <iostream>
using namespace std;

void quickSort(int arr[], int low, int high)
{
    if (low > high)
        return;

    int pivot = arr[high];
    int i = low;

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

    int temp = arr[i];
    arr[i] = arr[high];
    arr[high] = temp;

    quickSort(arr, low, i - 1);
    quickSort(arr, i + 1, high);
}

int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    quickSort(arr, 0, size - 1);

    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

## Sample Input - Output:



```
PS F:\All Codes\JavaScript> cd "f:\All Codes
5 11 12 22 25 34 64 90
PS F:\All Codes\JavaScript>
```

## Result Analysis / Discussion:

Quick Sort sorts an array by selecting a pivot element and placing smaller elements before it and larger elements after it. This process is repeated recursively for the subarrays. It is efficient, works in-place, and has an average time complexity of O(n log n).