

## Experiment No. 01

**Student Name: Rishant Shukla**

**Branch: MCA - CCD**

**Semester: I**

**Subject Name: Design and Analysis of Algorithms**

**UID: 22MCC20053**

**Section/Group: MCD-1/ Grp B**

**Date of Performance: 26<sup>th</sup> Sept 22**

**Subject Code: 22CAP-646**

### 1. Task to be done:

A) Sort the list by quick sort and write an algorithm for quick sort.

B) Explain divide and conquer and write an algorithm for merge sort.

### 2. Algorithm for experiment/practical:

(A)

```
//partition function which rearranges the array such that
// all the smaller elements are left to pivot element
// and all the larger elements are right to pivot element.

partition(arr[], lo, hi)
    pivot = arr[hi]
    PIndex = lo    // place for swapping
    for i := lo to hi - 1 do
        if arr[i] <= pivot then
            swap arr[PIndex] with arr[i]
            PIndex = PIndex + 1
    swap arr[PIndex] with arr[hi]
    return PIndex

// we divide the array into two subarrays around
// the pivot and recursively call for them separately.

quicksort(arr[], lo, hi)
    if lo < hi
        PIndex = partition(arr, lo, hi)
        quicksort(arr, lo, p-1)
        quicksort(arr, p+1, hi)
```

(B)

```
mergeSort(arr[],l,r)  //arr is array, l is left, r is right
{
    if(l<r)
    {
        midpoint = (l+r)/2
        mergeSort(arr,l,m)
        mergeSort(arr,m+1,r)
        merge(arr,l,m,r)
    }
}
```

### 3. Code for experiment/practical:

A)

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

void
swap(int * a, int * b)
{
    int
    t = *a;
    *a = *b;
    *b = t;
}

void
printArray(int
array[], int
size) {
    int
    i;
    for (i = 0; i < size; i++)
        cout << array[i] << " ";
    cout << endl;
}

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()
{
int
data[] = {44, 33, 11, 55, 77, 90, 40, 60, 99, 22, 88};
int
n = sizeof(data) / sizeof(data[0]);

cout << "Unsorted Array: \n";
printArray(data, n);
cout << "\n";
quickSort(data, 0, n - 1);

cout << "Sorted array in ascending order: \n";
printArray(data, n);
}
```

**B)**

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

void merge(int arr[], int l, int m, int r) {
    int i = l;
    int j = m + 1;
    int k = l;

    int temp[5];

    while (i <= m && j <= r) {
        if (arr[i] <= arr[j]) {
            temp[k] = arr[i];
            i++;
            k++;
        } else {
            temp[k] = arr[j];
            j++;
            k++;
        }
    }

    while (i <= m) {
        temp[k] = arr[i];
        i++;
        k++;
    }

    while (j <= r) {
        temp[k] = arr[j];
        j++;
        k++;
    }

    for (int p = l; p <= r; p++) {
        arr[p] = temp[p];
    }
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = (l + r) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

int main() {
    int myarray[5];
    //int arr_size = sizeof(myarray)/sizeof(myarray[0]);
```

```
int arr_size = 5;

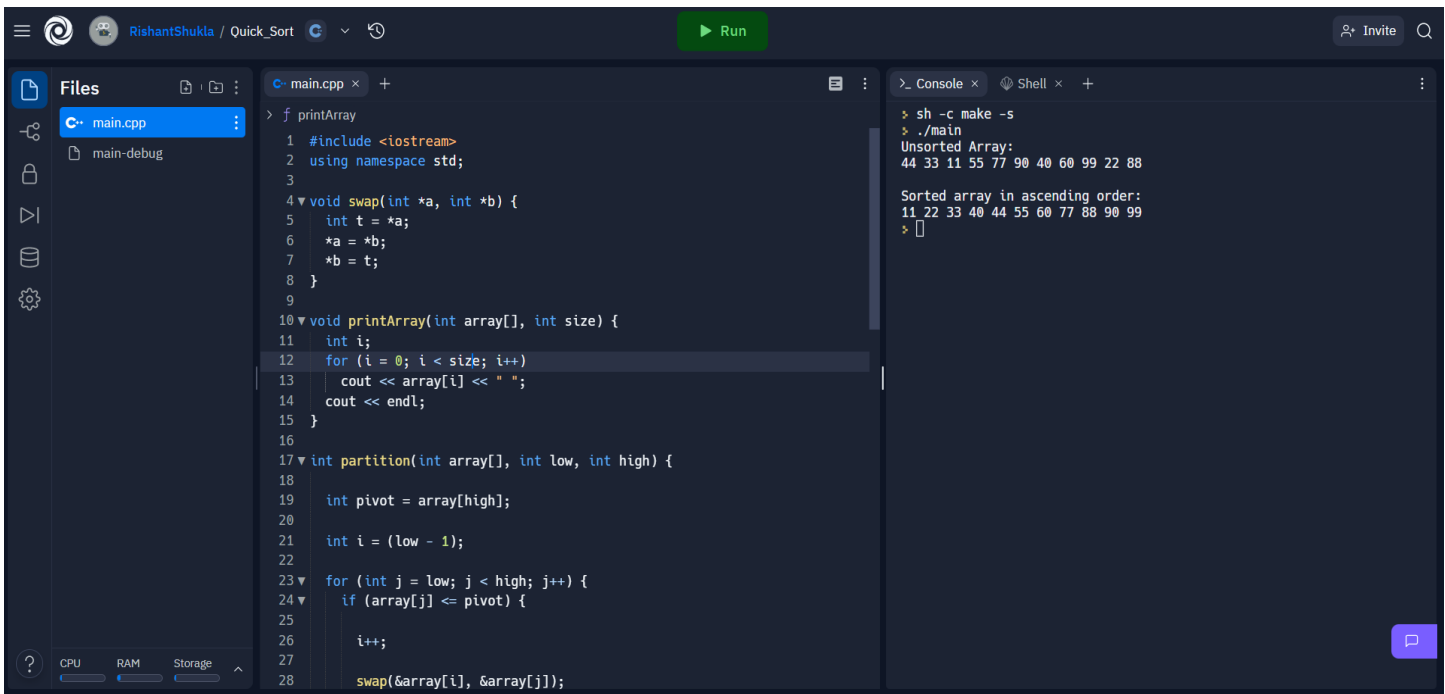
cout << "Enter 5 integers in any order: " << endl;
for (int i = 0; i < 5; i++) {
    cin >> myarray[i];
}
cout << "Before Sorting" << endl;
for (int i = 0; i < 5; i++) {
    cout << myarray[i] << " ";
}
cout << endl;
mergeSort(myarray, 0, (arr_size - 1));

cout << "After Sorting" << endl;
for (int i = 0; i < 5; i++) {
    cout << myarray[i] << " ";
}

return 0;
}
```

#### 4. Output:

A)



The screenshot shows a C++ IDE with a file named `main.cpp` and a console window. The code implements a merge sort algorithm. The console output shows the unsorted array and the sorted array in ascending order.

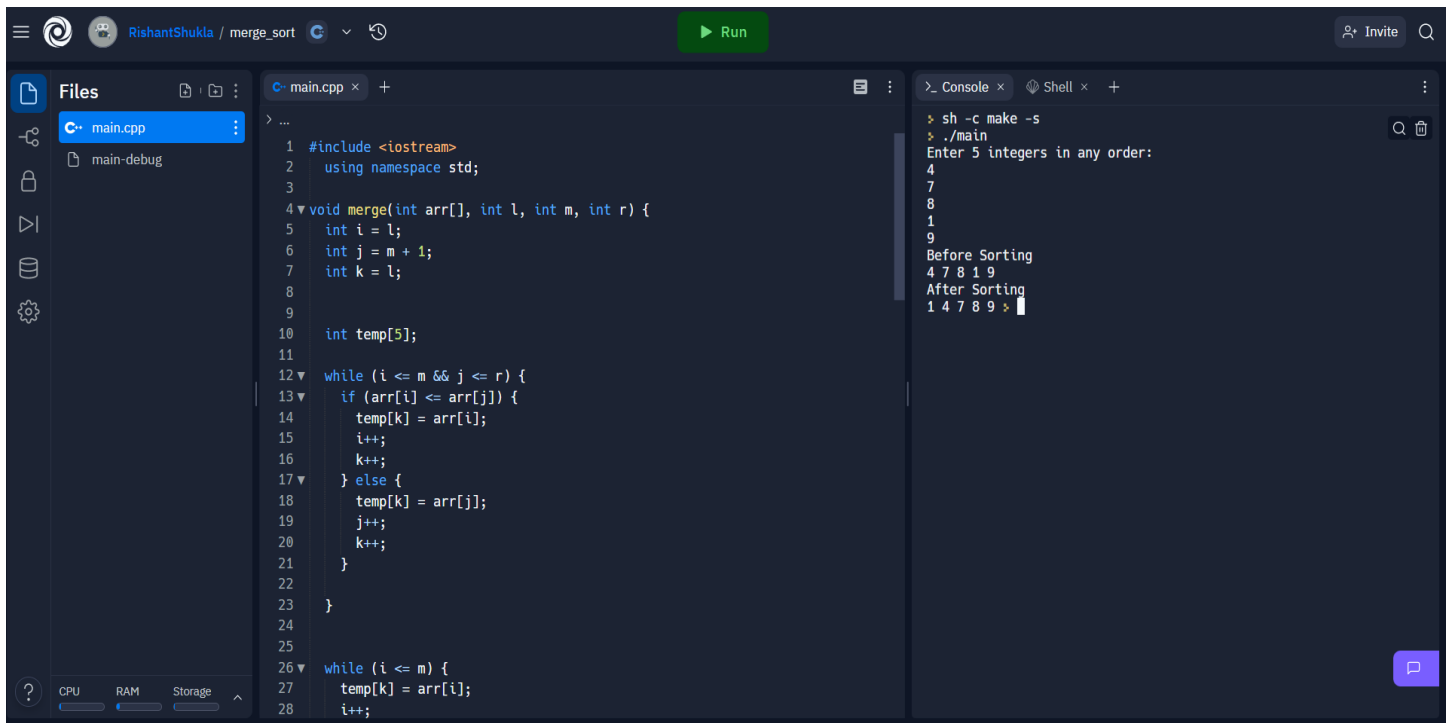
```
> f printArray
1 #include <iostream>
2 using namespace std;
3
4 void swap(int *a, int *b) {
5     int t = *a;
6     *a = *b;
7     *b = t;
8 }
9
10 void printArray(int array[], int size) {
11     int i;
12     for (i = 0; i < size; i++)
13         cout << array[i] << " ";
14     cout << endl;
15 }
16
17 int partition(int array[], int low, int high) {
18     int pivot = array[high];
19
20     int i = (low - 1);
21
22     for (int j = low; j < high; j++) {
23         if (array[j] <= pivot) {
24             i++;
25             swap(&array[i], &array[j]);
26         }
27     }
28     swap(&array[i], &array[high]);
29 }
```

Console Output:

```
> sh -c make -s
> ./main
Unsorted Array:
44 33 11 55 77 90 40 60 99 22 88

Sorted array in ascending order:
11 22 33 40 44 55 60 77 88 90 99
>
```

**B)**



```

1 #include <iostream>
2 using namespace std;
3
4 void merge(int arr[], int l, int m, int r) {
5     int i = l;
6     int j = m + 1;
7     int k = l;
8
9     int temp[5];
10
11     while (i <= m && j <= r) {
12         if (arr[i] <= arr[j]) {
13             temp[k] = arr[i];
14             i++;
15             k++;
16         } else {
17             temp[k] = arr[j];
18             j++;
19             k++;
20         }
21     }
22
23     while (i <= m) {
24         temp[k] = arr[i];
25         i++;
26     }
27     while (j <= r) {
28         temp[k] = arr[j];
29         j++;
30     }
31     for (i = l; i <= r; i++) {
32         arr[i] = temp[i - l];
33     }
34 }
35
36 int main() {
37     int arr[] = {4, 7, 8, 1, 9};
38     int l = 0, m = 2, r = 4;
39     merge(arr, l, m, r);
40     for (int i = 0; i < 5; i++) {
41         cout << arr[i] << " ";
42     }
43     return 0;
44 }

```

```

> sh -c make -s
> ./main
Enter 5 integers in any order:
4
7
8
1
9
Before Sorting
4 7 8 1 9
After Sorting
1 4 7 8 9

```

## Learning outcomes (What I have learned):

1. Learned to implement the quick sort and algorithm.
2. Learned to implement the merge sort and algorithm.

## Evaluation Grid:

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Demonstration and Performance		22