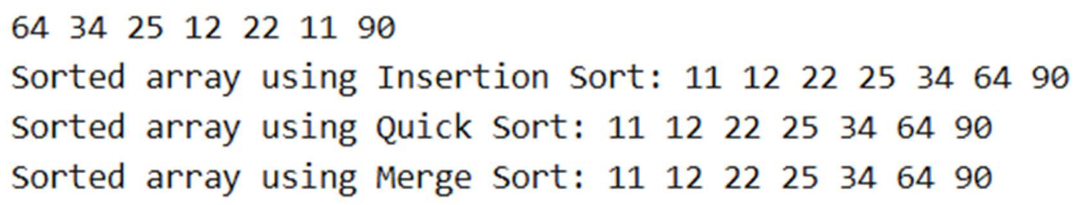
Experiment 9

SORTING:INSERTION SORT,QUICK SORT,MERGE SORT

```
#include <iostream>

#include <vector>


// Insertion Sort

void insertionSort(std::vector<int>& arr) {

    int n = arr.size();

    for (int i = 1; i < n; i++) {

        int key = arr[i];

        int j = i - 1;


        // Move elements of arr[0..i-1] that are greater than key

        while (j >= 0 && arr[j] > key) {

            arr[j + 1] = arr[j];

            j--;

        }

        arr[j + 1] = key;

    }

}


// Quick Sort

int partition(std::vector<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++;
            std::swap(arr[i], arr[j]);
        }
    }
    std::swap(arr[i + 1], arr[high]);
    return i + 1;
}

void quickSort(std::vector<int>& arr, int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);  // Before pi
        quickSort(arr, pi + 1, high); // After pi
    }
}

// Merge Sort
void merge(std::vector<int>& arr, int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    std::vector<int> L(n1), R(n2);

    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
```

```
}

void mergeSort(std::vector<int>& arr, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

// Utility function to print the array
void printArray(const std::vector<int>& arr) {
    for (int val : arr) {
        std::cout << val << " ";
    }
    std::cout << std::endl;
}

int main() {
    std::vector<int> arr1 = {64, 34, 25, 12, 22, 11, 90};
    printArray(arr1);
    std::vector<int> arr2 = arr1; // Copy for quick sort
    std::vector<int> arr3 = arr1; // Copy for merge sort

    // Insertion Sort
    insertionSort(arr1);

    std::cout << "Sorted array using Insertion Sort: ";

    printArray(arr1);


    // Quick Sort

    quickSort(arr2, 0, arr2.size() - 1);

    std::cout << "Sorted array using Quick Sort: ";

    printArray(arr2);


    // Merge Sort

    mergeSort(arr3, 0, arr3.size() - 1);

    std::cout << "Sorted array using Merge Sort: ";

    printArray(arr3);

    return 0;

}
```

OUTPUT:



```
64 34 25 12 22 11 90
Sorted array using Insertion Sort: 11 12 22 25 34 64 90
Sorted array using Quick Sort: 11 12 22 25 34 64 90
Sorted array using Merge Sort: 11 12 22 25 34 64 90
```