Name = Muhammad Waleed Sattar

SAP = 55700

Section = BS SE3-2

DSA LAB-12:

Task 1:

D: > DSA > DSA LAB > Lab 12 > Task_1.cpp > main()

```cpp
void printArray(const vector<int> &arr){
    for (int num:arr)
        cout<<num<<" ";
    }
    cout<<endl;
}

int main(){
    vector<int> arr={16,14,53,31,7,29,41,6,19};
    int n=arr.size();

    cout<<"Original Array: ";
    printArray(arr);

    clock_t start=clock();
    quickSort(arr, 0, n-1);
    clock_t end=clock();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\mwale> cd "d:\DSA\DSA LAB\Lab 12\" ; if ($?) { g++ Task_1.cpp -o Task_1 } ; if ($?) { .\Task_1 }
Original Array: 16 14 53 31 7 29 41 6 19
Sorted Array: 6 7 14 16 19 29 31 41 53
Time taken by quickSort: 0 Seconds
PS D:\DSA\DSA LAB\Lab 12>
```

Task 2:

```cpp
void measurePerformance(vector<int> arr) {
    double timeOriginal = double(end - start) / CLOCKS_PER_SEC;
    cout << "Time taken by Quick Sort (original): " << timeOriginal << " seconds" << endl;

    start = clock();
    quickSortRandom(arrCopy, 0, arrCopy.size() - 1);
    end = clock();
    double timeRandom = double(end - start) / CLOCKS_PER_SEC;
    cout << "Time taken by Quick Sort (random pivot): " << timeRandom << " seconds" << endl;
}

int main() {
    srand(time(0));

    int n = 10000;
    vector<int> arr(n);
    for (int i = 0; i < n; i++) {
        arr[i] = rand() % 100000;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\mwale> cd "C:\Users\mwale\AppData\Local\Temp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\temp
Comparing Quick Sort with Original and Random Pivot Selection on Array of Size 10000:
Time taken by Quick Sort (original): 0.003 seconds
Time taken by Quick Sort (random pivot): 0.003 seconds
PS C:\Users\mwale\AppData\Local\Temp>
```

Task 3:

```cpp
void measurePerformance(int size){
    double time_taken = double(end-start)/CLOCKS_PER_SEC;
    cout<<"Size: "<<size<<", Time taken: "<<time_taken<<" Seconds"<<endl;
}

int main(){
    int sizes[] = {100, 1000, 10000, 50000, 100000};
    int numSizes = sizeof(sizes) / sizeof(sizes[0]);

    for(int i = 0; i < numSizes; i++){
        measurePerformance(sizes[i]);
    }

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\mwale> cd "C:\Users\mwale\AppData\Local\Temp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) {
Size: 100, Time taken: 0 Seconds
Size: 1000, Time taken: 0 Seconds
Size: 10000, Time taken: 0.003 Seconds
Size: 50000, Time taken: 0.018 Seconds
Size: 100000, Time taken: 0.032 Seconds
PS C:\Users\mwale\AppData\Local\Temp>
```

Task 4:

```cpp
void measurePerformance(void (*sortFunc)(int[], int, int), int arr[], int size) {
    auto end = chrono::high_resolution_clock::now();
    chrono::duration<double> duration = end - start;
    cout << "Array Size: " << size << " - Time taken: " << duration.count() << " seconds\n";
}

int main() {
    const int sizes[] = {100, 1000, 10000};

    for (int size : sizes) {
        int* arr = new int[size];

        srand(time(0));
        for (int i = 0; i < size; i++) {
            arr[i] = rand() % 1000;
        }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\mwale> cd "C:\Users\mwale\AppData\Local\Temp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile }

Hybrid Quick Sort (with Insertion Sort for small subarrays):
Array Size: 100 - Time taken: 1.7e-05 seconds

Hybrid Quick Sort (with Insertion Sort for small subarrays):
Array Size: 1000 - Time taken: 0.0001721 seconds

Hybrid Quick Sort (with Insertion Sort for small subarrays):
Array Size: 10000 - Time taken: 0.0020186 seconds
PS C:\Users\mwale\AppData\Local\Temp>
```

Task 5:

```cpp
int main() {

    for (int size : sizes) {
        int* arrQuickSort = new int[size];
        int* arrMergeSort = new int[size];

        srand(time(0));
        for (int i = 0; i < size; i++) {
            int value = rand() % 1000;
            arrQuickSort[i] = value;
            arrMergeSort[i] = value;
        }

        cout << "\nTesting with Array Size: " << size << "\n";

        measurePerformance(quickSortRandomPivot, arrQuickSort, size, "Quick Sort (Random Pivot)");
        measurePerformance(mergeSort, arrMergeSort, size, "Merge Sort");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Quick Sort (Random Pivot) - Array Size: 100 - Time taken: 1.76e-05 seconds
Merge Sort - Array Size: 100 - Time taken: 9.91e-05 seconds

Testing with Array Size: 1000
Quick Sort (Random Pivot) - Array Size: 1000 - Time taken: 0.0002518 seconds
Merge Sort - Array Size: 1000 - Time taken: 0.0007254 seconds

Testing with Array Size: 10000
Quick Sort (Random Pivot) - Array Size: 10000 - Time taken: 0.0023225 seconds
Merge Sort - Array Size: 10000 - Time taken: 0.0098591 seconds
PS C:\Users\mwale\AppData\Local\Temp>
```

Task 6:

```
}

int main() {
    const int sizes[] = {100, 1000, 10000};

    for (int size : sizes) {
        int* arrQuickSort = new int[size];
        int* arrQuickSortRandom = new int[size];
        int* arrMergeSort = new int[size];
        int* arrInsertionSort = new int[size];

        srand(time(0));
        for (int i = 0; i < size; i++) {
            int value = rand() % 1000;
            arrQuickSort[i] = value;
            arrQuickSortRandom[i] = value;
            arrMergeSort[i] = value;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Quick Sort (Regular Pivot) - Array Size: 1000 - Time taken: 0.0001909 seconds
Quick Sort (Random Pivot) - Array Size: 1000 - Time taken: 0.0002152 seconds
Merge Sort - Array Size: 1000 - Time taken: 0.0008489 seconds
Insertion Sort - Array Size: 1000 - Time taken: 0.0009476 seconds

Testing with Array Size: 10000
Quick Sort (Regular Pivot) - Array Size: 10000 - Time taken: 0.0023303 seconds
Quick Sort (Random Pivot) - Array Size: 10000 - Time taken: 0.0024076 seconds
Merge Sort - Array Size: 10000 - Time taken: 0.0085271 seconds
Insertion Sort - Array Size: 10000 - Time taken: 0.104408 seconds
PS C:\Users\mwale\AppData\Local\Temp>
```