



RIPHAH

INTERNATIONAL UNIVERSITY

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()
34 void printArray(const vector<int> &arr){
35     for (int num:arr)
36         cout<<num<<" ";
37     }
38     cout<<endl;
39 }
40 }
41
42 int main(){
43     vector<int> arr={16,14,53,31,7,29,41,6,19};
44     int n=arr.size();
45
46     cout<<"Original Array: ";
47     printArray(arr);
48
49     clock_t start=clock();
50     quickSort(arr, 0, n-1);
51     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:

```
56 void measurePerformance(vector<int> arr) {
57     double timeOriginal = double(end - start) / CLOCKS_PER_SEC;
58     cout << "Time taken by Quick Sort (original): " << timeOriginal << " seconds" << endl;
59
60     start = clock();
61     quickSortRandom(arrCopy, 0, arrCopy.size() - 1);
62     end = clock();
63     double timeRandom = double(end - start) / CLOCKS_PER_SEC;
64     cout << "Time taken by Quick Sort (random pivot): " << timeRandom << " seconds" << endl;
65 }
66
67 int main() {
68     srand(time(0));
69
70     int n = 10000;
71     vector<int> arr(n);
72     for (int i = 0; i < n; i++) {
73         arr[i] = rand() % 100000;
74     }
75
76     measurePerformance(arr);
77     measurePerformance(arrCopy);
78 }
79
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  Code

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

```
56 void measurePerformance(int size){
57     double time_taken = double(end-start)/CLOCKS_PER_SEC;
58     cout<<"Size: "<<size<<" , Time taken: "<<time_taken<<" Seconds"<<endl;
59 }
60
61
62 int main(){
63     int sizes[] = {100, 1000, 10000, 50000, 100000};
64     int numSizes = sizeof(sizes) / sizeof(sizes[0]);
65
66     for(int i = 0; i < numSizes; i++){
67         measurePerformance(sizes[i]);
68     }
69
70     return 0;
71 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

```
49 void measurePerformance(void (*sortFunc)(int[], int, int), int arr[], int size) {
50     sortFunc(arr, 0, size - 1);
51     auto start = chrono::high_resolution_clock::now();
52     auto end = chrono::high_resolution_clock::now();
53     chrono::duration<double> duration = end - start;
54     cout << "Array Size: " << size << " - Time taken: " << duration.count() << " seconds\n";
55 }
56
57 int main() {
58     const int sizes[] = {100, 1000, 10000};
59
60     for (int size : sizes) {
61         int* arr = new int[size];
62
63         srand(time(0));
64         for (int i = 0; i < size; i++) {
65             arr[i] = rand() % 1000;
66         }
67
68         // ... (sort function call) ...
69     }
70 }
```

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:

```
76 int main() {
77     const int sizes[] = {100, 1000, 10000};
78
79     for (int size : sizes) {
80         int* arrQuickSort = new int[size];
81         int* arrMergeSort = new int[size];
82
83         srand(time(0));
84         for (int i = 0; i < size; i++) {
85             int value = rand() % 1000;
86             arrQuickSort[i] = value;
87             arrMergeSort[i] = value;
88         }
89
90         cout << "\nTesting with Array Size: " << size << "\n";
91
92         measurePerformance(quickSortRandomPivot, arrQuickSort, size, "Quick Sort (Random Pivot)");
93         measurePerformance(mergeSort, arrMergeSort, size, "Merge Sort");
94     }
95 }
```

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:

```
114 }
115
116 int main() {
117     const int sizes[] = {100, 1000, 10000};
118
119     for (int size : sizes) {
120         int* arrQuickSort = new int[size];
121         int* arrQuickSortRandom = new int[size];
122         int* arrMergeSort = new int[size];
123         int* arrInsertionSort = new int[size];
124
125         srand(time(0));
126         for (int i = 0; i < size; i++) {
127             int value = rand() % 1000;
128             arrQuickSort[i] = value;
129             arrQuickSortRandom[i] = value;
130             arrMergeSort[i] = value;
131             arrInsertionSort[i] = value;
132         }
133     }
134
135     // Timing for size 1000
136     QuickSort (Regular Pivot) - Array Size: 1000 - Time taken: 0.0001909 seconds
137     QuickSort (Random Pivot) - Array Size: 1000 - Time taken: 0.0002152 seconds
138     Merge Sort - Array Size: 1000 - Time taken: 0.0008489 seconds
139     Insertion Sort - Array Size: 1000 - Time taken: 0.0009476 seconds
140
141     // Timing for size 10000
142     Testing with Array Size: 10000
143     QuickSort (Regular Pivot) - Array Size: 10000 - Time taken: 0.0023303 seconds
144     QuickSort (Random Pivot) - Array Size: 10000 - Time taken: 0.0024076 seconds
145     Merge Sort - Array Size: 10000 - Time taken: 0.0085271 seconds
146     Insertion Sort - Array Size: 10000 - Time taken: 0.104408 seconds
147
148     PS C:\Users\mwale\AppData\Local\Temp>
```