Quick Sort in C++

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

int medianOfThree(int arr[], int l, int h) {
    int mid = l + (h - l) / 2;
    if (arr[l] > arr[mid]) swap(arr[l], arr[mid]);
    if (arr[l] > arr[h]) swap(arr[l], arr[h]);
    if (arr[mid] > arr[h]) swap(arr[mid], arr[h]);
    return mid;
}

int partition(int arr[], int l, int h) {
    int medianIndex = medianOfThree(arr, l, h);
    swap(arr[l], arr[medianIndex]);  // Move median to start as pivot

    int pivot = arr[l];
    int left = l + 1;
    int right = h;

    while (left <= right) {
        while (left <= right && arr[left] < pivot) left++;
        while (left <= right && arr[right] > pivot) right--;

        if (left <= right) {
            swap(arr[left], arr[right]);
            left++;
            right--;
        }
    }

    swap(arr[l], arr[right]);  // Put pivot in correct place
    return right;
}

void rquicksort(int arr[], int l, int h) {
    if (l < h) {
        int pivot = partition(arr, l, h);
        rquicksort(arr, l, pivot - 1);
        rquicksort(arr, pivot + 1, h);
    }
}

int main() {
    int arr[] = {24, 97, 40, 67, 88, 85, 15};
```

Here's a **dry run** of your Quicksort code **in tabular form** for the input:

int arr[] = {24, 97, 40, 67, 88, 85, 15};

We'll trace:

- Recursive calls
- Chosen pivot (via median-of-three)
- Partitioning process
- Array state after each step

🔁 Step-by-Step Dry Run Table:

Step	Subarray (l to h)	Median-of-Three	Pivot	Final Pivot Index	Array After Partition
1	arr[0..6] = {24,97,40,67,88,85,15}	40 (mid=2)	40	2	{24,15,40,67,88,85,97}
2	arr[0..1] = {24,15}	15 (mid=0)	15	0	{15,24,40,...}
3	arr[1..1] = {24}	-	-	-	(Base case, already sorted)
4	arr[3..6] = {67,88,85,97}	85 (mid=4)	85	4	{...,67,85,88,97}
5	arr[3..3] = {67}	-	-	-	(Base case)
6	arr[5..6] = {88,97}	88 (mid=5)	88	5	{...,67,85,88,97} (already sorted)

int n = sizeof(arr) / sizeof(arr[0]);
 	
rquicksort(arr, 0, n - 1);
 	
cout << "Sorted array: ";
for (int i = 0; i < n; i++) {
cout << arr[i] << " ";
}
cout << endl;
 	
return 0;
}
	🧠 **Final Sorted Array:**
	
	{15, 24, 40, 67, 85, 88, 97}

15, 24, 40, 67, 85, 88, 97	