**📋 Initial List:**

ini

Copy code

delivery\_times = [45, 30, 60, 20, 50, 35, 25]

**⚙️ Quick Sort Explanation:**

Quick Sort is a **divide-and-conquer** algorithm. Here's how it works:

1. Choose a **pivot** (commonly the last element).
2. Partition the list into:
   * Elements **less than** the pivot.
   * Elements **greater than** the pivot.
3. Recursively apply the same process to the sublists.

**🧮 Step-by-Step (simplified trace):**

**Step 1:** Choose pivot = 25  
Partition:

* Less than 25: [20]
* Greater than 25: [45, 30, 60, 50, 35]
* Equal to pivot: [25]

Now recursively sort:  
[20] + [25] + quick\_sort([45, 30, 60, 50, 35])

**Step 2:** Sorting [45, 30, 60, 50, 35], pivot = 35  
Partition:

* Less: [30]
* Greater: [45, 60, 50]
* Pivot: [35]

Now: [30] + [35] + quick\_sort([45, 60, 50])

**Step 3:** Sorting [45, 60, 50], pivot = 50  
Partition:

* Less: [45]
* Greater: [60]
* Pivot: [50]

Now: [45] + [50] + [60]

**🧩 Combine All Steps:**

css

Copy code

[20] + [25] + [30] + [35] + [45] + [50] + [60]

**✅ Final Sorted List:**

csharp

Copy code

[20, 25, 30, 35, 45, 50, 60]

**C Program:**

#include <stdio.h>

// Function to swap two elements

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// Partition function

int partition(int arr[], int low, int high) {

int pivot = arr[high]; // Choosing the last element as pivot

int i = low - 1; // Index of smaller element

for (int j = low; j < high; j++) {

if (arr[j] < pivot) {

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]); // Place pivot in correct position

return (i + 1);

}

// Quick Sort function

void quickSort(int arr[], int low, int high) {

if (low < high) {

// Partition index

int pi = partition(arr, low, high);

// Recursively sort elements before and after partition

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

// Function to print the array

void printArray(int arr[], int size) {

for (int i = 0; i < size; i++)

printf("%d ", arr[i]);

printf("\n");

}

// Main function

int main() {

int delivery\_times[] = {45, 30, 60, 20, 50, 35, 25};

int n = sizeof(delivery\_times) / sizeof(delivery\_times[0]);

printf("Original delivery times:\n");

printArray(delivery\_times, n);

quickSort(delivery\_times, 0, n - 1);

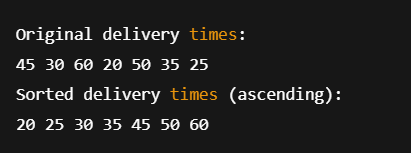
printf("Sorted delivery times (ascending):\n");

printArray(delivery\_times, n);

return 0;

}

**OUTPUT:**

****