Create a Header File (`sorting_algorithms.h`)

```
#ifndef SORTING_ALGORITHMS_H
#define SORTING_ALGORITHMS_H

void bubbleSort(int arr[], int n);
void insertionSort(int arr[], int n);
void selectionSort(int arr[], int n);
void quicksort(int arr[], int low, int high);

#endif // SORTING_ALGORITHMS_H
```

2. Create the Implementation File (`sorting_algorithms.c`)

```
#include "sorting_algorithms.h"
#include <stdio.h>

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int min_idx = i;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
        int temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

int partition(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++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}

void quicksort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quicksort(arr, low, pi - 1);
        quicksort(arr, pi + 1, high);
    }
}
```

3. Create the Testing File (`testing.c`)

```c
#include <stdio.h>
#include "sorting_algorithms.h"

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr1[] = {64, 25, 12, 22, 11};
    int arr2[] = {64, 25, 12, 22, 11};
    int arr3[] = {64, 25, 12, 22, 11};
    int arr4[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr1)/sizeof(arr1[0]);

    printf("Original array: \n");
    printArray(arr1, n);

    bubbleSort(arr1, n);
    printf("Sorted array with Bubble Sort: \n");
    printArray(arr1, n);

    insertionSort(arr2, n);
    printf("Sorted array with Insertion Sort: \n");
    printArray(arr2, n);

    selectionSort(arr3, n);
    printf("Sorted array with Selection Sort: \n");
    printArray(arr3, n);

    quicksort(arr4, 0, n-1);
    printf("Sorted array with Quicksort: \n");
    printArray(arr4, n);

    return 0;
}
```

Explanation

1. **Header File (`sorting_algorithms.h`):**
    - This file contains function prototypes for the sorting algorithms.
2. **Implementation File (`sorting_algorithms.c`):**
    - Contains the implementation of the sorting algorithms: bubble sort, insertion sort, selection sort, and quicksort.
    - Also includes a helper function `partition` used by quicksort.
3. **Testing File (`testing.c`):**
    - Contains the `main` function to test the sorting algorithms.
    - Includes a `printArray` function to display the contents of the array before and after sorting.
    - Tests all sorting algorithms with the same initial array to compare results.