```c
/*
*
* Implement bubble sort
* Peter Strawn
* CS50 AP
*
*/

#include <cs50.h>
#include <stdio.h>

#define LENGTH 10

int main(void)
{
    // declare array of 10 numbers
    int unsorted[LENGTH] = { 9, 7, 3, 4, 2, 8, 1, 6, 0, 5 };

    // print unsorted array
    printf("Unsorted\n");
    for (int i = 0; i < LENGTH; i++)
    {
        printf("unsorted[%d]: %d\n", i, unsorted[i]);
    }
    printf("\n");

    /* bubble sort */
    // set swaps to any non-zero value
    int swaps = -1;

    // track total number of swaps (just for fun)
    int totalSwaps = 0;

    // repeat until no more swaps (i.e. list is sorted)
    while (swaps != 0)
    {
        // set swaps to zero
        swaps = 0;

        // iterate through numbers in unsorted array
        for (int i = 0; i < LENGTH - 1; i++)
        {
            // if a number is greater than the number on its right, swap their positions
            if (unsorted[i] > unsorted[i+1])
            {
                int temp = unsorted[i];
                unsorted[i] = unsorted[i+1];
                unsorted[i+1] = temp;

                // increment swaps variable for each swap
                swaps++;
                totalSwaps++;
            }
        }
    }

    // print sorted array
    printf("Sorted\n");
    for (int i = 0; i < LENGTH; i++)
    {
        printf("unsorted[%d]: %d\n", i, unsorted[i]);
    }
    printf("\nTotal number of swaps: %d\n", totalSwaps);
}
```

```c
/*
*
* Implement selection sort
* Peter Strawn
* CS50 AP
*
*/

#include <cs50.h>
#include <stdio.h>

#define LENGTH 10

int main(void)
{
    // declare array of 10 numbers
    int unsorted[LENGTH] = { 9, 7, 3, 4, 2, 8, 1, 6, 0, 5 };

    // print unsorted array
    printf("Unsorted\n");
    for (int i = 0; i < LENGTH; i++)
    {
        printf("unsorted[%d]: %d\n", i, unsorted[i]);
    }
    printf("\n");

    /* selection sort */
    // track total number of swaps (just for fun)
    int totalSwaps = 0;

    // iterate through numbers in unsorted array
    // stop at second-to-last index value since that will make j the last index value
    for (int i = 0; i < LENGTH - 1; i++)
    {
        // set index of minimum value to i
        int min = i;

        // iterate through rest of numbers in list to update minimum value of necessary
        for (int j = i + 1; j < LENGTH; j++)
        {
            if (unsorted[j] < unsorted[min])
            {
                min = j;
            }
        }

        // swap positions of i and minimum if i isn't already the minimum
        if (min != i)
        {
            int temp = unsorted[min];
            unsorted[min] = unsorted[i];
            unsorted[i] = temp;
            totalSwaps++;
        }
    }

    // print sorted array
    printf("Sorted\n");
    for (int i = 0; i < LENGTH; i++)
    {
        printf("unsorted[%d]: %d\n", i, unsorted[i]);
    }
    printf("\nTotal number of swaps: %d\n", totalSwaps);
}
```

```c
/*
*
* Implement insertion sort
* Peter Strawn
* CS50 AP
*
*/

#include <cs50.h>
#include <stdio.h>

#define LENGTH 10

int main(void)
{
    // declare array of 10 numbers
    int unsorted[LENGTH] = { 9, 7, 3, 4, 2, 8, 1, 6, 0, 5 };

    // print unsorted array
    printf("Unsorted\n");
    for (int i = 0; i < LENGTH; i++)
    {
        printf("unsorted[%d]: %d\n", i, unsorted[i]);
    }
    printf("\n");

    /* insertion sort */
    // track total number of swaps (just for fun)
    int totalSwaps = 0;

    // iterate through numbers in the unsorted list (i.e. start at index 1)
    for (int i = 1; i < LENGTH; i++)
    {
        // set second item in the array as the first unsorted item
        // by default, unsorted[0] is already a sorted array of one item
        int sortValue = unsorted[i];

        // set index of second item in the array as the first unsorted item
        int sortIndex = i;

        // move sortValue down the list so long as two conditions are met:
        // 1. sortIndex does not go below 1 since it's being compared with the number to its left
        // 2. sortValue is less than the value to its left in the array
        while (sortIndex > 0 && unsorted[sortIndex - 1] > sortValue)
        {
            int temp = unsorted[sortIndex - 1];
            unsorted[sortIndex - 1] = unsorted[sortIndex];
            unsorted[sortIndex] = temp;
```

```
                sortIndex--;
                totalSwaps++;
            }
        }

        // print sorted array
        printf("Sorted\n");
        for (int i = 0; i < LENGTH; i++)
        {
            printf("unsorted[%d]: %d\n", i, unsorted[i]);
        }
        printf("\nTotal number of swaps: %d\n", totalSwaps);
    }
```