CMSC21 Lab Exercise 15 – Arrays and pointers

Traversing an array using pointer arithmetic doesn't make programming with arrays any more easier. However, using pointers to refer to arrays can prove useful in another way: producing a "sub-array" of a given array for use in recursive functions.

Consider the following recursive displayArray function:

```
void displayArray(int* arr, int length) {
  if (length == 0)  {
    printf("\n");
  } else {
    printf("%d ", *arr);
    displayArray(arr + 1, length - 1);
  }
}
```

Take a few minutes to study this function. Can you explain what's happening? Type it in and run it, passing an integer array. Is the output what you expect?

Similar to the function above, create the following recursive functions, using pointers to refer to int arrays. Use pointer arithmetic (and shorter lengths) to split the array up into sub-arrays.

1. Create a function that recursively finds the smallest value in the array. Follow this algorithm:

    1. If the length of an array is 1, then the first element is the smallest value
    2. Otherwise, the smallest value is the minimum of either of the following values:
        a) the first element
        b) the smallest value of the rest of the array

2. Create a function that recursively finds the smallest value in the array. Follow this algorithm:

    1. If the length of an array is 1, then the first element is the smallest value
    2. Otherwise, the smallest value is the minimum of either of the following values:
        a) the smallest value of the first half of the array
        b) the smallest value of the second half of the array

3. **(Optional)** Create a function that recursively determines whether an array is a palindrome. An array is palindrome if it is of length 1, or if values at both ends are equal, and the middle sub-array is a palindrome.