5) (10 pts) ALG (Sorting)

For this question, implement the (very slow) sorting algorithm described below:

1. Randomly choose two distinct array indexes, i and j. (If i and j are equal, choose again.)
2. If array[min(i,j)] > array[max(i,j)], swap the two values.
3. Check if the array is sorted. If it's not, go back to step 1. If it is, return.

Recall that the function call `rand()` returns a random non-negative integer, so `rand()%n` will equal a random integer in between 0 and n-1.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int min(int a, int b) {if (a < b) return a; return b;}
int max(int a, int b) {if (a > b) return a; return b;}

void randomSort(int* array, int length) {

    while (1) {

        int i = 0, j = 0;
        while (i == j) {
            i = rand()%length;
            j = rand()%length;
        }
        int minI = min(i,j);
        int maxI = max(i,j);

        if (array[minI] > array[maxI]) {
            int temp = array[minI];
            array[minI] = array[maxI];
            array[maxI] = temp;
        }

        int sorted = 1;
        for (i=0; i<length-1; i++)
            if (array[i] > array[i+1])
                sorted = 0;

        if (sorted) break;
    }

}
```

Grading: 3 pts picking random indices, 3 pts performing swap, if necessary, 3 pts check at end, 1 pt stopping/breaking if the array is sorted