Arrays as Pointers

This exercise will give you a chance to work with arrays using the pointer syntax. You'll find a source file, sortList.c on the course homepage in moodle and under the following path in AFS:

/afs/eos.ncsu.edu/courses/csc/csc230/common/www/sturgill/exercise10

This program is a partial implementation of a selection sort, using lots of pointer operations and pointer arithmetic. Fill in the missing parts of the swap(), findSmallest() and printList() functions to complete the program. For the findSmallest() and printList() functions, I've put constraints on what code you can add. This is to help make sure you get practice working with pointers. Instead of manipulating the array using the usual array syntax, we will access it via pointers to its elements.

As described in the source file, you get to complete three functions:

- swap( ) : this is the usual function used to swap the contents of two integers passed in by address. You could probably find a copy of this function somewhere in the lecture notes or the example programs, but why not try to write it yourself, just for the practice.
- findSmallest() : Given an array (pointer to int) and a length, return a pointer to the smallest value in the array. In this function, you will use a pointer (rather than an index) to keep up with a particular element of an array.
- printList() : Given an array (pointer to int) and a length, print out the contents of the array. This function will let you traverse an array via a pointer (rather than an index).

When your program is working, it should produce output like the following (don't worry about writing a fencepost loop to suppress printing a trailing space):

74 166 250 273 441 545 659 710 808 879 924 931

When you're done submit your completed sortList.c program via WolfWare Classic.