CS 271 and 462

PA 5 - Programming Assignment 5

Overview of This Assignment

4 Files:

- 2 C source files
- 1 header file
- 1 makefile

You must upload the files to a CS Linux host, make, then run the executable on Linux.
Once you have finished and tested the programs, you will upload them to the PA 5 assignment in Canvas.

C Programming Skills That You Are Practicing in This Assignment

1. Become proficient at writing and using makefiles.
2. Become proficient in working with pointers in C.
3. Master the process of swapping.
4. Develop an understanding of the selection sort and insertion sort algorithms.

makefile - do this first!

- Include an "all" target that builds an executable named pa5 (no extension).
- Write separate targets for pa5, pa5.o, and sortingFunctions.o.
- Include a "clean" target that removes all object files (all files ending with .o).

pa5.c - the test program

1. Create a file named pa5.c. Copy the following code into your program. This is the initial version of your "test program". You will modify this file as you go through the process of writing and testing the functions in this assignment.

   The comments are there to show examples of what you might do to test your other functions.

```
#include <stdio.h>
#include <stdlib.h>
#include "sortingFunctions.h"
#define ARRAYSIZE 10

int main (void) {

   // dynamically allocate memory space for an array
   int * arrayPtr = (int *) malloc (ARRAYSIZE * sizeof(int));

   // fill the array with random integers

   // print the array, 10 elements per line

   // sort the array using selection sort and print the return value

   // print the array, 10 elements per line
```

```
    // fill the array again with random integers

    // print the array

    // sort with insertion sort and print the return value

    // print the array

}
```

2. Create a file called sortingFunctions.h Write a preprocessor wrapper (following convention for constant name) and then insert the following prototypes.

```
int selectionSort ( int * const data, int size );
int insertionSort ( int * const data, int size );
void swap ( int * num1, int * num2 );
void fillArray( int * const data, int size, int min, int max );
void neatPrint ( int * const data, int size, int numPerLine, int fieldSize );
```

3. Create a file called sortingFunctions.c Write the implementation of the five functions.

```
int selectionSort ( int * const data, int size );
```

- sorts the array using the selection sort algorithm

  Caution: there are a lot of programs out there on the web that claim to be selection sort written in C. The class syllabus prohibits copying code from web sites. A grade of zero will be assigned for this assignment if the TA or instructor sees evidence that you have copied code.

- You cannot use the bracket [ ] notation to access array elements. You have to use pointer/offset notation. If you use [ ], the grader will enter a score of zero for this function.

- The return value is a count of the number of comparisons that are made. Specifically, you should count the number of times that the program executes an if statement condition that compares the values of two array elements.

```
int insertionSort ( int * const data, int size );
```

- sorts the array using the insertion sort algorithm

  Caution: there are a lot of programs out there on the web that claim to be insertion sort written in C. The class syllabus prohibits copying code from web sites. A grade of zero will be assigned for this assignment if the TA or instructor sees evidence that you have copied code.

- You cannot use the bracket [ ] notation to access array elements. You have to use pointer/offset notation. If you use [ ], the grader will enter a score of zero for this function.

- The return value is a count of the number of comparisons that are made. Specifically, you should count the number of times that the program executes an if statement condition or a while loop condition that compares the values of two array elements.

```
void swap ( int * num1, int * num2 );
```

- Exchanges the contents of the two memory locations.

```
void fillArray( int * const data, int size, int min, int max );
```

- Fills the array elements with integers from min to max (inclusive).

- You cannot use the bracket [ ] notation to access array elements. You have to use pointer/offset notation. If you use [ ], the grader will enter a score of zero for this function.

```
void neatPrint ( int * const data, int size, int numPerLine, int fieldSize );
```

- Prints the array elements in nice, neat columns using the parameters numPerLine and fieldSize to control the layout.

- You cannot use the bracket [ ] notation to access array elements. You have to use pointer/offset notation. If you use [ ], the grader will enter a score of zero for this function.

4. Compile, test, and debug your program as needed.

5. Submit 4 files. Do not zip or tar the files.

   - pa5.c
   - sortingFunctions.h
   - sortingFunctions.c
   - makefile (You will need to copy this to a file named makefile.c so you can submit it.)