ECE 71 – Engineering Computations in C

Professor Kriehn – Fall 2017

Code Due By: Midnight on Thursday, Nov 16, 2017
Writeup Due By: Class on Mon/Tue, Nov 20/21, 2017

HOMEWORK #32 – Fun with Pointers

Write a program that prompts the user to input 3 integer values into the computer. Write a function that re-orders the values in the three integer variables such that the values occur in ascending order.

Specifications:
Use the following function prototype for your sorting function:

```
void ptrSort(int *a, int *b, int *c);
```

Use the following three variables to store your numbers in the `main()` function:

```
int x, y, z;
```

Use the following three pointers to point to the three variables in the `main()` function:

```
int *i, *j, *k;
```

After you have your integer variables and pointers declared, prompt the user to enter 3 numbers (1 number at a time). Read in the numbers into the variables using your pointers, and call your `ptrSort()` function – whose purpose is to sort the numbers so that x contains the smallest number, y contains the middle number, and z contains the largest number. Then print the results to the screen using the pointer variables. **You MUST use pointers to perform ALL operations relating to scanning, sorting, and printing the numbers.**

If you execute the program, the following information should be displayed:

```
~> main.o
Please enter the first number: 3
Please enter the second number: 1
Please enter the third number: 2

The sorted numbers are: 1, 2, and 3.

~> main.o
Please enter the first number: -15
Please enter the second number: -30
Please enter the third number: 212

The sorted numbers are: -30, -15, and 212.
~>
```

HOMEWORK #33 – Dynamic Arrays

Write a program that asks the user to enter an integer to create an array of size *n*. Use the value to create an *n*-element dynamic double array by allocating memory appropriately and hooking up a pointer to the array. Then allow the user to enter a double value into each array entry. Loop through the array, calculate the average, and print the result. Delete the memory allocated to the dynamic array before exiting the program.

Specifications:

You may not use vectors to solve the problem. Allocate and deallocate memory with the `new` and `delete` commands, respectively, and use a pointer to point to the allocated array of memory locations.

Format the output to 2 values after the decimal point.

If you execute the program, the following information should be displayed:

```
~> main.o
Enter size of the dynamic array: 3

Enter Element 1: 6.0
Enter Element 2: 7.9
Enter Element 3: 8.2

The average is 7.37

~> main.o
Enter size of the dynamic array: 10

Enter Element 1: 6.0
Enter Element 2: 7.9
Enter Element 3: 8.2
Enter Element 4: -3.4
Enter Element 5: -2.7
Enter Element 6: 7.2
Enter Element 7: 1.7
Enter Element 8: 0.67
Enter Element 9: 12.7
Enter Element 10: -4.9

The average is 3.34
```