CS 271 and 462

PA 3 - Programming Assignment 3

See the link "Creating a Makefile" in the Resources Module.

Reminder: The gcc flag for compile only (and do not create an executable) is `-c`. Don't use `-c` and `-o` together in the same gcc command.

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 3 assignment in Canvas.

C Programming Skills That You Are Practicing in This Assignment

1. Writing functions in C.
2. Writing and using a makefile to expedite the compiling and linking phases of development.
3. Improving your documentation and style skills.
4. Improving the readability of your programs.
5. Increasing your debugging skills to include syntax errors in functions and function calls.

Reminder: If you submit a program that contains a syntax error, you will receive a grade of zero for that program.

Grading Rubric

The rubric for this assignment has 25% of the assignment score allocated to documentation, style, and readability. Make sure you read the rubric before you submit.

As you go...Backup Your Work

Each time you work on an assignment, make a backup of your files. Some techniques for backup:

- copy the files onto a USB drive
- email the files to yourself
- upload the files to cloud storage (OneDrive, GoogleDocs, Dropbox, etc)

Warning: If you have Linux installed on your own computer and you have the GNU gcc compiler, you may try using your own computer but you should still test your files on a CS host computer before submitting.

1. Make a new directory for this lab. Place all of the files you create for PA 3 in that directory.

2. Create a makefile. There is a separate instruction sheet for this.
   - The "all" target must produce the executable **pa3**, all lowercase, no extension.
   - Required targets: all, pa3, pa3.o, pa3functions.o
   - The clean target is not required.

3. Create a header file named pa3functions.h. In this file, place the following lines:

```
#ifndef PA3FUNCTIONS
#define PA3FUNCTIONS
        here is where your function prototypes go
#endif
```

4. Create a source file named pa3functions.c   In this file, place the implementations of the following three functions:
   - Write a function named **diagonal**. The function should accept 1 parameter, an integer named **number**. The function will use the parameter value to print a pattern similar to the one shown below. (Note: this example output is for a parameter value of 5.)

```
     5
    4
   3
  2
 1
0
```

```
Incremental Programming

It is highly recommended
that you write and test only
one function at a time.
```

There are 5 spaces before the number 5, 4 spaces before the number 4, etc.

- Write a function named alphabet. The function should accept 2 parameters: a char named **letterCase** and an integer named **lettersPerLine.**

  If letterCase contains 'U', the function should print the uppercase alphabet.

  If letterCase contains 'L', the function should print the lowercase alphabet.

  If letterCase contains something other than 'U' or 'L', the function should print an error message and return.

  The parameter lettersPerLine determines how many letters are printed on each line of output. If lettersPerLine is less than or equal to zero, the the function should print an error message and return.

- Write a function named **randomNumber**. The function should accept 2 parameters: an integer named **min** and an integer named **max**. The return type of the function is int.

  If min > max, the function should print an error message and return 0.

  Otherwise, the function should generate and return a random integer between min and max (inclusive).

5. Create a source file named pa3.c. In this file, place the main function as follows:

```
The items I listed below are required tests.  You may add
additional test calls if you wish.

Remember to print meaningful messages with the output.
The messages must say which function is being tested and
what parameter values are being used.
```

```
// header comments
#include "pa3functions.h"
#include <stdio.h>
int main (void) {
   // Testing the diagonal function
   // Input an integer from the user.  The program must use a while loop
   // to insure that the number is between 1 and 9 (inclusive). Just keep
   // prompting them until they enter a number in that range.
   // Then, call the diagonal function with the user's input as the parameter.

   // Testing the alphabet function
   // Call the alphabet function with parameters 'U' and 5.
   // Call the alphabet function with parameters 'L' and 8.
   // Call the alphabet function with parameters 'X' and 10.
```

```
    // Testing the randomNumbers function
    // Call the randomNumber function 100 times (in a loop) with parameters 100
    // and 200.  Print the 100 numbers, 10 numbers per line, using a field
    // width of 5.
    // Call the randomNumber function once with invalid parameters for min
    // and max.

}
```

6. "make" pa3. If the makefile works correctly, run the executable.
7. Debug, make again, ... repeat until all 3 functions work correctly.

Submit 4 files:

1) makefile (You must submit a copy of the makefile named makefile.c.)
2) pa3functions.h
3) pa3functions.c
4) pa3.c

Make sure that you select the correct files and that all programs are uploaded before you submit.

From the syllabus: "It is your responsibility to upload the correct files and to ensure that the uploaded files contain the work that you want to have graded."