S2021 CST8234 – C Programming

Lab 1: After Environment Setup; test your programming basics

Programming Exercise Objective:

In this lab assessment, you will demonstrate what you have learned with respect to

- main() and other user defined functions
- Macros
- Pre-processor directives, variables, and libraries usage
- C compiler practice
- How to run C code effectively
- Input validation and proper error message

Program #1:

Write a small program: {(00_Numbers.c)} that:-

1. Prints the numbers from 1 to 100
2. If the number is a multiple of three, it should print instead "*I'm a multiple of 3!*"
3. If the number is a multiple of five, it should print instead "*I'm a multiple of 5!*"
4. If the number is a multiple of three and five, it should print instead "*I'm a multiple of 3 && 5!*"

This program should not take you more than 10 minutes to write.

The following demonstrates the execution of the program:

```
#./00_Numbers
…
8
9 I'm multiple of 3!!!
10 I'm multiple of 5!!!
11
12 I'm multiple of 3!!!
13
14 15 I'm multiple of 3 && 5!!!
```

SAMPLE TEST OUTPUT: 00_Numbers

Program #2:

Write a small C program: *pretty_phone.c* that:-

1. Read an integer number from the command line using *scanf( )*
2. If the number is a valid 7-digit phone number, it should print the phone number formatted: XXX-YYYY
   a. where XXX is the 3-digit central office code
   b. where YYYY is the 4-digit subscriber line
3. If the number is not a valid 7-digit phone number, it should print an appropriate error message when…
   a. the number is less than 7-digits
   b. the number is greater than 7-digits
   c. the central office code begins with a zero (0) or one (1)
4. A phone number is valid when:
   a. 7-digits in length
   b. the central office code cannot begin with zero (0) nor one (1)
5. Repeat until a value of *0*.

The following demonstrates the execution of the program:

```
kraken:hurdleg_L1 hurdleg$ ./pretty_phone
Enter a phone number: 7274723      [normal case]
727-4723
Enter a phone number: 1000000      [min phone#]
Invalid central office code: 1
Enter a phone number: 2000000      [true min phone#]
200-0000
Enter a phone number: 9999999      [max phone#]
999-9999
Enter a phone number: 10000000     [max+1 = too many]
Invalid phone number: too many digits
Enter a phone number: 0234567      [scanf() ignores leading 0s]
Invalid phone number: too few digits
Enter a phone number: 02345678
234-5678
Enter a phone number: 0            [exit with success]
kraken:hurdleg_L1 hurdleg$ echo $? [echo return code]
0
```

SAMPLE TEST OUTPUT: pretty_phone

Program #3:

Write a small C program: *bin2dec.c* that:-

1. Read a binary number (just *0* and *1*) from the command line using *scanf( )*.
2. Prints the decimal number equivalent to the binary number entered.
3. Repeat until a value of *0*.

```
kraken:hurdleg_L1 hurdleg$ ./bin2dec
Enter a binary number: 1100
The decimal equivalent of 1100 is 12
Enter a binary number: 11101
The decimal equivalent of 11101 is 29
Enter a binary number: 10
The decimal equivalent of 10 is 2
Enter a binary number: 0
kraken:hurdleg_L1 hurdleg$ echo $?
0
kraken:hurdleg_L1 hurdleg$
```

SAMPLE TEST OUTPUT: *bin2dec*

Requirements:

1) Create a folder called **algonquinUserID1_L1** (e.g., "mynam00123_L1"). Do all of your work in this folder, and when complete, submit the zipped folder as per the "Lab Instructions" posted on Brightspace.
2) You must implement all possible user input data verification.
3) Each function must have header comments that explain what it does and describe/explain its inputs (if any) and return value (if any) and the code must be properly commented.

Marking:

This assignment is out of 20 points:

- □ 10 points for Program #2
  - 05 for coding correctness (i.e., correct results)
  - 05 for demonstration during scheduled lab
- □ 10 points for Program #3
  - 05 for coding correctness (i.e., correct results)
  - 05 for demonstration during scheduled lab

Submission:

- Code (.c file) → Compressed Zipfolder
- Screenshots (terminal: a) gcc -ansi -pedantic -Wall and b) input(s)/output(s)) → Compressed Zipfolder
- Demonstration is mandatory. A missed demonstration will result in 50% deduction.