C Bitwise Calculator

Purpose: The purpose of this assignment is to give you the opportunity to help familiarize you with some of the basic C language functions, the binary representation of numbers as well as binary operators and arithmetic. **This is an individual assignment, not a team assignment**. In this assignment you will copy and modify the program you built for lab 2 (C Calculator).

Project setup on your server (5%)

1. Login to your server.
2. Change directories to be in the directory `/home/your_user_name/CS_1337/assignments`
3. Copy your `lab_2` directory to create a `lab_3` using the command

   ```
   cp -r lab_2 lab_3
   ```

4. Change directories to be in the directory `/home/your_user_name/CS_1337/assignments/lab_3`
5. Renaming your source file to be `bitcalc.c` (case-sensitive) using the command

   ```
   mv ccalc.c bitcalc.c
   ```

6. Using good programming practices (comments, frequent testing, good variable naming, etc.), implement a program that satisfies all of the requirements listed below.
7. When you submit this assignment, the submission form will ask for the path to your assignment on your server. You should submit the path exactly as it is returned from the `pwd` command. It should look something like `/home/your_user_name/CS_1337/assignments/lab_3`

Program correctness requirements (80%)

Program correctness will be determined by the correct functionality of the program as well as adherence to format specifications.

Prompt user for operator (collectively 5%)

8. The program will first prompt the user for an operation. The possible operations are |, &, and ^ (or, and, and xor). If q is entered as the operation, then the program should quit.

9. If any other operation is input, the program should print the message “Please enter |, &, ^, or q” and reissue the original prompt on the next line.
10. The code that you use for validating input should be extracted into a function.

Prompt user for the operand count (collectively 5%)

11. After a valid operation has been entered, prompt the user for an integer. (This will represent the number of integers to include in the calculation.)
12. The user must enter a valid integer greater than one. If the user's input is invalid, you should print the message "Please enter an integer greater than 1" and reissue the prompt. I suggest considering the use of the sscanf function for this purpose. We will not test your programs with integers that overflow the maximum integer size.
13. The code that you use for validating input should be extracted into a function and should handle erroneous input of arbitrary length.

Prompt user for hexadecimal operands

14. (5%) Finally, prompt the user for *n* **hexadecimal** integers, where *n* is the number you received as input from the previous prompt. The integers must be formatted in hexadecimal and must have a maximum of 8 hexadecimal digits. Integers that are input with less than 8 digits should be assumed to have leading zeros.
15. (10%) Any invalid input should result in the message “Please enter an 8-digit hexadecimal integer”, followed by the prompt for that integer on the next line. There should be one prompt per integer (in the manner shown below).
16. (5%) You should use arrays to store these values.
17. (5%) The code that you use for validating input should be extracted into a function and should handle erroneous input of arbitrary length.

Perform the calculation

18. (25%) After receiving valid input, the program should then perform the operation on the numbers in the form `firstnum operator secondnum [operator thirdnum etc]`. The expression should be evaluated from left to right. The program should then print back the entire expression being computed (including the operands, operation, and result) in **both hexadecimal and binary**.
19. (5%) Your program should use a switch statement in this portion of the assignment.
20. (10%) The binary values should be padded so that there are 32 total digits, grouped into clusters of 8 digits.

Example program execution (with correct formatting)

(5%) A major goal of this assignment is to test your attention to detail. **Therefore, you must follow the output format shown above exactly**. Prompts in your program must exactly match the prompts shown above. The same number of spaces that are used above must be used in your program. In particular, there should be one space after the prompt before the user input. There should be a space between each operand and operator. There will be no spaces on the right-hand side of any line after the output. Indentation of operation results should use a tab character (plus space as needed). Your program will be diffed against expected output, and formatting errors will result in a loss of points.

Below is the format for a sample operation, where red text indicates user-entered values:

```
Enter operation: |
Enter number of integers: 2
Enter integer 1: 000000FF
Enter integer 2: 30C01101

Hexadecimal operation:
       000000FF
     | 30C01101
     = 30C011FF

Binary operation:
       00000000 00000000 00000000 11111111
     | 00110000 11000000 00010001 00000001
     = 00110000 11000000 00010001 11111111

Enter operation: q
```

Here is another output of a possible execution of the program:

```
Enter operation: @dfa
Please enter |, &, ^, or q
Enter operation: &
Enter number of integers: 1
Please enter an integer greater than 1
Enter number of integers: 3
Enter integer 1: DF5383
Enter integer 2: Z0238984
Please enter an 8-digit hexadecimal integer
Enter integer 2: 908070605040
Please enter an 8-digit hexadecimal integer
Enter integer 2: C5204
Enter integer 3: F13FB
```

```
Hexadecimal operation:
       00DF5383
     & 000C5204
     & 000F13FB
     = 000C1200

Binary operation:
       00000000 11011111 01010011 10000011
     & 00000000 00001100 01010010 00000100
     & 00000000 00001111 00010011 11111011
     = 00000000 00001100 00010010 00000000

Enter operation: q
```

Other requirements (15%)

21. (5%) This project should have a makefile. The first rule in your makefile should compile the code for this assignment into an executable called `bitcalc`.
22. (5%) Before submitting this assignment, you need to push your code to your GitHub repo (see the EC2 tutorial section entitled "Everyday Git commands"). You should include a README.md file (case sensitive) in the project directory to explain briefly to the user how to compile (i.e., call `make`) and execute your program.
23. (5%) Your code will be evaluated for quality. In particular, you should observe the following:
    1. **Variable, class, and function naming conventions**. Each language has its own conventions. The most important thing is consistency and using meaningful names.
    2. **Clear and concise comments**. Functions should have comments. In addition, each block of code should be prefaced with a comment.
    3. **Indentations**. The most important thing is readability and consistency.