Coding Assignment 2

1. The format and content of the output is not a suggestion – it is the specification given to you to follow so please follow it exactly. Points will be lost for not following the specification. Formatting the output to look exactly the same is part of exercises in this assignment.

2. Please note that part of the rubric is ***how*** you coded the program in addition to outputting valid responses. Be sure to review the rubric before submitting your code.

3. Make sure your name and student id are in a comment in the first line of your program.

4. Name your program `Code2_studentid.c` and name the zip file `Code2_studentid.zip` for submission to Canvas.

Pseudocode

A `#define` called `BITS` should be set at the top of the program. It should be set to 8 when the program is submitted. This `define` should be used throughout the entire program when setting/using array sizes/max element. This `define` will also be used in the output to print `8-bit` vs `16 bit`. Your program should function whether the `define` is set to 8 or to 16. Part of the grading process will be to change the define from 8 to 16 and to recompile your program to see if your program still runs properly. See sample output.

`main()`

Print instructions (see sample output below)

Prompt for and store both numbers and the operator. Use only one `scanf()` to store all three values.

While either number is less than 0 or more than 255, continue to prompt for input until a valid number is entered (hint – use a while loop). See sample output below.

Call function `ConvertDecimalToBinary()` to convert the first number to binary.

Call function `ConvertDecimalToBinary()` to convert the second number to binary.

If the entered operator is an allowed operator, then convert the result to binary and print the decimal result and binary result as seen in the sample code.

`ConvertDecimalToBinary()`

Return type : `void`

Parameters :

`int` containing the decimal value to be converted

`char` array (hint : the array is passed empty from `main()` and this function fills it so when the function finishes, the array back in `main()` will contain the values added in the function).

This function will use a method of decimal to binary conversion called “Divide in Half, Ignore the Remainder”. Please watch the following video for a demonstration of the method.

https://youtu.be/XdZqk8BXPwg

Create a local `int` array. This array will store the result of each divide by 2 which will be accomplished using bitshifting instead of division to divide the number in half. Use a bitmask to determine if an array element if odd (1) or even (0).

Using a `for` loop, loop over the `int` array and write each element into the `char` array that was passed in. Hint : keep in mind what the ASCII value is for the number zero when writing the `int` array element into the `char` array. If you store the number 65 in a `char`, it will be 'A' so if you want to store the number 0 in a `char` array, you will need to …?

HINT : make sure your `char` arrays are one bigger than the number of `BITS` so that you have room for the null terminator so that `%s` prints correctly.

```
[frenchdm@omega CA2]$ a.out

Bitwise Calculator

Enter two base 10 values with a bitwise operator to see the decimal result
and the binary result.  The format is

FirstNumber BitwiseOperator SecondNumber

For example, enter the expression

2 & 3

This calculator can used with &, |, ^, << and >>

Please note that the spaces between numbers and operator is essential
and the two entered values must be between 0 and 255

Enter expression 2 & 3

In base 10...

2 & 3 = 2


In 8-bit base 2...

  00000010
&
  00000011
  ========
  00000010

[frenchdm@omega CA2]$ a.out

Bitwise Calculator

Enter two base 10 values with a bitwise operator to see the decimal result
and the binary result.  The format is

FirstNumber BitwiseOperator SecondNumber

For example, enter the expression
```

```
2 & 3

This calculator can used with &, |, ^, << and >>

Please note that the spaces between numbers and operator is essential
and the two entered values must be between 0 and 255

Enter expression 2 | 3

In base 10...

2 | 3 = 3


In 8-bit base 2...

  00000010
|
  00000011
  ========
  00000011

[frenchdm@omega CA2]$ a.out

Bitwise Calculator

Enter two base 10 values with a bitwise operator to see the decimal result
and the binary result.  The format is

FirstNumber BitwiseOperator SecondNumber

For example, enter the expression

2 & 3

This calculator can used with &, |, ^, << and >>

Please note that the spaces between numbers and operator is essential
and the two entered values must be between 0 and 255

Enter expression 2 ^ 3

In base 10...

2 ^ 3 = 1


In 8-bit base 2...

  00000010
^
  00000011
  ========
  00000001
```

```
[frenchdm@omega CA2]$ a.out

Bitwise Calculator

Enter two base 10 values with a bitwise operator to see the decimal result
and the binary result.  The format is

FirstNumber BitwiseOperator SecondNumber

For example, enter the expression

2 & 3

This calculator can used with &, |, ^, << and >>

Please note that the spaces between numbers and operator is essential
and the two entered values must be between 0 and 255

Enter expression 2 << 3

In base 10...

2 << 3 = 16


In 8-bit base 2...

00000010 << 3

00010000

[frenchdm@omega CA2]$ a.out

Bitwise Calculator

Enter two base 10 values with a bitwise operator to see the decimal result
and the binary result.  The format is

FirstNumber BitwiseOperator SecondNumber

For example, enter the expression

2 & 3

This calculator can used with &, |, ^, << and >>

Please note that the spaces between numbers and operator is essential
and the two entered values must be between 0 and 255

Enter expression 2 >> 3

In base 10...

2 >> 3 = 0
```

```
In 8-bit base 2...

00000010 >> 3

00000000

[frenchdm@omega CA2]$ a.out

Bitwise Calculator

Enter two base 10 values with a bitwise operator to see the decimal result
and the binary result.  The format is

FirstNumber BitwiseOperator SecondNumber

For example, enter the expression

2 & 3

This calculator can used with &, |, ^, << and >>

Please note that the spaces between numbers and operator is essential
and the two entered values must be between 0 and 255

Enter expression 100 >> 3

In base 10...

100 >> 3 = 12


In 8-bit base 2...

01100100 >> 3

00001100

[frenchdm@omega CA2]$ a.out

Bitwise Calculator

Enter two base 10 values with a bitwise operator to see the decimal result
and the binary result.  The format is

FirstNumber BitwiseOperator SecondNumber

For example, enter the expression

2 & 3

This calculator can used with &, |, ^, << and >>

Please note that the spaces between numbers and operator is essential
```

```
and the two entered values must be between 0 and 255

Enter expression 2 * 3


Operator * is not supported by this calculator

[frenchdm@omega CA2]$
```