W04 Task Submission (Loops, Functions, and Bitwise Operators)

You should watch the videos under Instructor Videos for this module before completing these tasks. You may need to perform additional research to solve these tasks. You are encouraged to use the Support Forum to get help with these tasks. After completing your tasks, please submit your original C source files. You do not need to submit files you did not write. **In each of your programs, use comments to explain what the code does.**

1. Write a program that reads **a string from** standard input and sends to standard output the hexadecimal values of the **input** characters, each on its own line. **(Hint: printf can display hexadecimal numbers for you if you give it the right specifier!)**

Example input/output:

```
Enter a string: gtfj
67
74
66
6A
```

2. **Write a function that displays a number in a binary representation. This function takes one parameter: an unsigned int which represents the number to convert. This function uses printf to display the binary value.**

```
void display_binary(unsigned int num);
```

You may use any method you can devise to do this conversion. Here's one way to think about this problem. Each bit in an unsigned int can either be on (1) or off (0). You can use the '&' bitwise operator to determine if a particular bit is on or off. You can test if a bit is on by using the expression "x & (1 << n)" where x is the number you're converting and n is the bit number you're testing (which for a 4-byte unsigned int will range from 31 down to 0). If the value of this expression is nonzero you display a '1', otherwise you display a '0'.

You may include leading zeroes and spaces between octets (sets of 8 bits) in the converted number **if you wish**.

In your main function, you will ask the user for the value to convert and call your display_binary() function which will display the converted number.

For example, given the input number 3, the output could be something like one of these:

```
00000000000000000000000000000011
11
00000011
00000000 00000000 00000000 00000011
```

3. Use the bitwise operators & ^ | ~ << >> using values and operations of your choice to perform bitwise operations and display the results to the user using the **display_binary** function you already wrote. Use each operator at least once.