Bitwise Questions

Notes:

- This Assignment needs the application of concepts of operators, control flow constructs, macro, functions.
- Write functions wherever necessary.
- All programs should follow ***C-11 standard*** and ***GESL coding guidelines.***
- Use ***-Wall*** and ***-Wstrict-prototypes*** options while compiling to arrest any warnings.

Conventions:

- ***p*** – bit position, ***s*** – source bit position, ***d*** – destination bit position, ***n*** – number of bits, ***num*** – number, ***snum*** – source number, ***dnum*** – destination number. • Position (***p***) starts with 0 from the right. For e.g.,

  p : 7 6 5 4 3 2 1 0
  num : 1 1 1 0 0 1 0 0

- Right-adjusted means from given position, continue right. In the above example, extracting 3 bits (right adjusted) from pos 4 returns 001.
- Leading means left side and trailing means right side. In the above example,
  Count of leading set bits – 3 (3 leading 1's from left), count of leading cleared bits – 0 (as leading bit starts with 1), count of trailing set bits – 0 (as trailing bit starts with 0) and count of trailing cleared bits – 2 (2 trailing 0's from right).

1. Write a function:
   unsigned int swap_bits_within (unsigned int num, unsigned int s, unsigned int d) that returns ***num*** with bits ***s*** and ***d*** swapped, leaving other bits unchanged.

2. Write a function:
   int swap_bits_between (unsigned int snum, unsigned int dnum,
   unsigned int s, unsigned int d);
   that swaps bit values between ***s*** in number ***snum*** and ***d*** in number ***dnum,*** leaving other bits unchanged. The function returns 0 if successful, else -1.

3. Write a function:
   unsigned int copy_bits (unsigned int snum, unsigned int dnum,
   unsigned int n,
   unsigned int s, unsigned int d) that returns ***dnum*** with ***n*** bits (right adjusted) copied from ***s*** in ***snum*** to ***d*** in ***dnum***.

4. Write the following functions:
   a. Toggle even bits: ***unsigned int toggle_even_bits (unsigned int num);***
   b. Toggle odd bits: ***unsigned int toggle_odd_bits (unsigned int num);***
   which will toggle all the even bits and all the odd bits respectively.

5. Write a macro:
   #define test_set_bit (num, p) that test and set a bit position ***p*** in a number.

6. Write the following functions:
   a. ***unsigned int left_rotate_bits (unsigned int num, unsigned int n)*** that returns the value of ***num*** rotated to the left by ***n*** positions.
   b. ***unsigned int right_rotate_bits (unsigned int num, unsigned int n)*** that returns the value of ***num*** rotated to the right by ***n*** positions.

7. Write the following functions:
    a. ***unsigned int count_set_bits (unsigned int num)***
    b. ***unsigned int count_clear_bits (unsigned int num)***

    that returns the count of number of bits set and number of bits cleared in ***num*** respectively.

8. Write the following functions:
    a. ***unsigned int count_leading_set_bits (unsigned int num)*** returns the number of leading set bits
    b. ***unsigned int count_leading_clear_bits (unsigned int num)*** returns the number of leading cleared bits
    c. ***unsigned int count_trailing_set_bits (unsigned int num)*** returns the number of trailing set bits
    d. ***unsigned int count_trailing_clear_bits (unsigned int num)*** returns the number of trailing cleared bits

9. Write macros for the following, using bitwise operations to:
    a. find maximum and minimum of two numbers.
    b. clear right most set bit/s in a number.
    c. clear left most set bit/s in a number.
    d. set right most cleared bit/s in a number.
    e. set left most cleared bit/s in a number.
    f. set bit/s from bit position ***s*** to bit position ***d*** in a number and clear all the rest.
    g. clear bit/s from bit position ***s*** to bit position ***d*** in a number and set all the rest.
    h. toggle bit/s from bit position ***s*** to bit position ***d*** in a number.

10. Write a function:

    unsigned int set_bits (unsigned int dnum, unsigned int p,
    unsigned int n, unsigned int snum)

    that returns ***dnum*** with the ***n*** bits that begin at position ***p*** set to the right most ***n*** bits of ***snum,*** leaving the other bits unchanged.

11. Write a function:

    unsigned int invert_bits (unsigned int snum, unsigned int p, unsigned int n) that returns ***snum*** with ***n***-bit field (right-adjusted) that begin at position ***p*** inverted, leaving others unchanged.

12. Write a macro:

    unsigned int get_bits (unsigned int snum, unsigned int p, unsigned int n) that returns ***n***-bit field (left-adjusted) of ***snum*** that begins at position ***p***.