Task 1: Bit Manipulation

1. Description

The task contains bits manipulation such as setting bit, clearing bit, clear bit, reversing bits order and more.

2. Program flow

The user shall first enter the operation they wants, then the program shall ask the user to enter the number to perform the operation on.

After performing the operations, the program shall print the input and output numbers in both decimal and binary representation.

3. Operations

The program shall allow multiple operations as follows:

Note: all examples described here are supposed to be on input numbers of type (unsigned char) and size of 1 byte (8 bits), your program should operate on inputs of type (unsigned int) and size of 4 bytes (32 bits).

- **(0) Clear Bit:**

  If the user enters (*0*) for the operation, then you will ask them to enter number and bit position in the same line separated by space.

  The program shall clear the selected bit in the binary representation of this number.

  For example, input number is 25 $(00011001)_2$ and the user enters bit position 4, means that 4th bit shall be cleared (set to 0). Note that counting starts from bit 0. So, the output shall be 9 $(00001001)_2$.

  If the entered bit is already 0, the program shall keep it 0.

  For example, input number is 13 $(00001101)_2$ and the user enters bit position 6, means that output shall be 13 $(00001101)_2$.

- **(1) Set Bit:**

  If the user enters (*1*) for the operation, then you will ask them to enter number and bit position in the same line separated by space.

  The program shall set the selected bit in the binary representation of this number.

For example, input number is 25 $(00011001)_2$ and the user enters bit position 2, means that 2nd bit shall be set (set to 1). Note that counting starts from bit 0. So, the output shall be 29 $(00011101)_2$.

If the entered bit is already 1, the program shall keep it 1.

For example, input number is 13 $(00001101)_2$ and the user enters bit position 0, means that output shall be 13 $(00001101)_2$.

- **<u>(2) Toggle Bit:</u>**

If the user enters (*2*) for the operation, then you will ask them to enter number and bit position in the same line separated by space.

The program shall toggle the selected bit in the binary representation of this number.

For example, input number is 25 $(00011001)_2$ and the user enters bit position 4, means that 4th bit shall be toggled (here it's 1, so set it to 0). So, the output shall be 9 $(00001001)_2$.

For example, input number is 13 $(00001101)_2$ and the user enters bit position 5, means that 2nd bit shall be toggled (here it's 0, so set it to 1). So, the output shall be 45 $(00101101)_2$.

- **<u>(3) Get Bit:</u>**

If the user enters (*3*) for the operation, then you will ask them to enter number and bit position in the same line separated by space.

The program shall get the selected bit in the binary representation of this number.

For example, input number is 25 $(00011001)_2$ and the user enters bit position 4, means that the output is the 4th bit value (here it's 1).

For example, input number is 13 $(00001101)_2$ and the user enters bit position 5, means that the output is the 4th bit value (here it's 0).

- **<u>(4) Rotate Bits:</u>**

If the user enters (*4*) for the operation, then you will ask them to enter number.

The program shall rotate the bits in the binary representation of this number.

For example, input number is 25 $(00011001)_2$, means that the output is 152 $(10011000)_2$. Note that bits are reversed (first became last and 2nd first became 2nd last and so on).

- **(5) Flip Bits:**

  If the user enters (*5*) for the operation, then you will ask them to enter number.

  The program shall flip the bits in the binary representation of this number.

  For example, input number is 25 $(00011001)_2$, means that the output is 230 $(11100110)_2$. Note that bits are flipped (zeros became ones and ones became zeros).

- **(6) Number of Ones:**

  If the user enters (*6*) for the operation, then you will ask them to enter number.

  The program shall get the number of ones in the binary representation of this number.

  For example, input number is 25 $(00011001)_2$, means that the output is 3 (There are 3 ones in the binary representation).

- **(7) Max Number of Consecutive Ones:**

  If the user enters (*7*) for the operation, then you will ask them to enter number.

  The program shall get the max number of consecutive ones in the binary representation of this number.

  For example, input number is 103 $(01100111)_2$, means that the output is 3 (There are 3 consecutive ones and 2 consecutive ones so, max value is 3).

- **(8) First Position of One:**

  If the user enters (*8*) for the operation, then you will ask them to enter number.

  The program shall get the first position of one in the binary representation of this number.

  For example, input number is 103 $(01100111)_2$, means that the output is 0 (First one at position 6).

- **(9) Last Position of One:**

  If the user enters (*9*) for the operation, then you will ask them to enter number.

  The program shall get the last position of one in the binary representation of this number.

  For example, input number is 103 $(01100111)_2$, means that the output is 6 (Last one at position 6).

4. Input Format

First you will ask the user to enter the operation, then:

For operations *0,1,2,3* you will ask them to enter a number and bit position in the same line, for example:

```
Please enter the operation: 2

Please enter the number and the bit position: 25 4
```

For operations *4,5,6,7,8,9* you will ask them to enter a number, for example:

```
Please enter the operation: 7

Please enter the number: 152
```

5. Output Format

Output format should be as following for each operation:

Operation *0,1,2,4,5*:

```
Please enter the operation: 0

Please enter the number and the bit position: 25 4

-----------

Input number in decimal is: 25

Input number in binary is: 00011001

Output number in decimal is: 9

Output number in binary is:  00001001
```

Operation *3*:

```
Please enter the operation: 3

Please enter the number and the bit position: 25 2

-----------

Input number in decimal is: 25

Input number in binary is: 00011001

Bit value is: 0
```

Operation *6,7,8,9*:

```
Please enter the operation: 8

Please enter the number: 25
```

```
-----------

Input number in decimal is: 103

Input number in binary is: 01100111

Output is: 6
```

6. Notes

- The program shall keep running till infinity, after printing the output to the user the program shall ask them again to enter new operation and so on. In case the user enters *10* for the operation, means that they need to exit the application and the program no longer ask them to enter the new operation.
- Your perfect program shall cover all corner cases and wrong inputs from the user to tell them that the input is invalid.
- Your program shall be divided into two files: main.c and bits_manipulation.c