Lab objective

The goal of this lab is to understand bitwise operations and their use:

Step 1) To begin with, let's create the following C++ program and place it in a file called main.cpp

```cpp
#include <iostream>
using namespace std;

void main()
{
    system("pause");
}
```

Step 2: Let's try to understand the built in data types available in C++, and how they are implemented in our compiler (Visual Studio C++ or others). Copy and paste the following code above the line “System()” in your main() function.

```cpp
cout << "====Understanding your Compiler and Datatypes====" << endl;
cout << "Size of char: " << sizeof(char) << endl;
cout << "Size of unsigned char: " << sizeof(unsigned char) << endl;
cout << "Size of unicode widechar: " << sizeof(wchar_t) << endl;
cout << "Size of int: " << sizeof(int) << endl;
cout << "Size of float: " << sizeof(float) << endl;
cout << "Size of double: " << sizeof(double) << endl;
```

Now compile and run the program to see the results:

```
==Understanding your Compiler and Datatypes==
Size of char: 1
Size of unsigned char: 1
Size of unicode widechar: 2
Size of int: 4
Size of float: 4
Size of double: 8
```

Step 3: Now, let's continue by creating a character, placing a certain bit pattern in it, and they try to display that character using different methods. Place the following code above the “System()” in your main() function.

```cpp
unsigned char aChar;
aChar = 'A';
cout << "--------- aChar ---------" << endl;
//std::cout.flags ( std::ios::showbase );
cout << "char format(C++) = " << char(aNum) << endl;
cout << "Dec format(C++) = " << int(aChar) << endl;
```
cout << "Octal format(C++) = " << std::oct << int(aChar) << endl;
cout << "HEX format(C++) = " << std::hex << int(aChar) << endl;
cout << "BIN format(C++) = " << std::bitset<8>(aChar) << endl; // Need to include <bitset>
cout << "---------------------------------" << endl;

Now compile and run the program to see the results. [however remember to include the <bitset>]

--- aChar ------
char format(C++) = A
Dec format(C++) = 0x41
Octal format(C++) = 0101
HEX format(C++) = 0x41
BIN format(C++) = 01000001

Step 3: As you can see we are able to view the character in many different formats. Specially interesting is our ability to view the bit pattern in a given character. That requires special library functions that may not be available in all compilers, so below will create our own small function to display the bits in an unsigned character. Copy and past the following function to your program.

```cpp
string char_to_binary(unsigned char value)  
{
    char theResult[128];
    unsigned char mask = 0x80;
    int i;

    for (i = 0; i < sizeof(value) * 8; i++) {
        if ((value & mask) == 0)
            theResult[i] = '0';
        else
            theResult[i] = '1';
        mask >>= 1;
    }
    theResult[i] = '\0';
    string result = theResult;
    return(result);
}
```

Make sure to include a forward declaration for it. Also make sure to include the heading for string manipulation, as the function uses a string object.

Now, place the following code above "System()" in your main() function

```cpp
    cout << "Binary format = " << char_to_binary(aChar) << endl;
cout << "---------------------------------" << endl;
```

Now compile and run the program to see the results. Note that our function produces the same bit pattern!
This small function can be very useful to us as we learn more about Bitwise operation.

Now, start by copying the following code segments (on segment at a time) into your program, then run it to see, examine and understand the results before proceeding to the next segment.

```cpp
cout << "===============Understanding Bitwise Shift Operations===============" << endl;
unsigned char clearChar;
clearChar = 0x01;
cout << "0x01 in Binary format   = " << char_to_binary(clearChar) << endl;
clearChar = (clearChar << 2);
cout << "Shift left 2 bits       = " << char_to_binary(clearChar) << endl;
clearChar = (clearChar >> 1);
cout << "Shift right 1 bit       = " << char_to_binary(clearChar) << endl;
clearChar = (clearChar << 6);
cout << "Shift left 6 bits       = " << char_to_binary(clearChar) << endl;

cout << "==================== OR Operation=========================
 clearChar = 0x01;
cout << "0000 0001 | 00001100 = " << (clearChar | 0x0C) << endl;
cout << "Hex:                   " << std::hex << (clearChar | 0x0C) << endl;
cout << "Binary Result:         " << char_to_binary(clearChar | 0x0C) << endl;

cout << "===================== AND Operation=========================
 clearChar = 0xff;
cout << "1111 1111 & 00001100 = " << (clearChar & 0x0C) << endl;
cout << "Hex:                   " << std::hex << (clearChar & 0x0C) << endl;
cout << "Binary Result:         " << char_to_binary(clearChar & 0x0C) << endl;

cout << "===================== XOR Operation=======================
 clearChar = 0x09;
cout << "0000 1001 ^ 0000 1100 = " << char_to_binary(clearChar ^ 0x0C) << endl;

cout << "================= Double XOR Operation=====================
 clearChar = 0x05; // a non-zero number
cout << "~0000 1001 = " << char_to_binary(~clearChar) << endl;
clearChar = 0x00; // a zero

cout << "===================== NOT Operation========================
 clearChar = 0x09;
cout << "~0000 1001 = " << char_to_binary(~clearChar) << endl;
```

Your final output should look similar to the screen shot below.
Understanding Bitwise Shift Operations

0x01 in Binary format  = 00000001
Shift left 2 bits      = 0000100
Shift right 1 bit     = 0000010
Shift left 6 bits     = 10000000

OR Operation

0000 0001 | 00001100 = 0xd
Hex: 0xd

Binary Result: 00001101

AND Operation

1111 1111 & 00001100 = 0xc
Hex: 0xc

Binary Result: 00001100

XOR Operation

0000 1001 ^ 0000 1100 = 00000101

Double XOR Operation

Back to the original char:
00001001 ^ 0000 1100 ^ 0000 1100 = 00001001

NOT Operation

~0000 1001 = 00000000
~0000 0000 = 00000001

Complement Operation

~0000 1001 = 11110110