Bit Shifting

CS 10A – BOOLEAN ALGEBRA PART 2

Bit Shifting

- Bit shifting is the act of moving a sequence of bits by a specified number of places in order to change its value.
  - Bit shift left: 0b00110011 << 1 == 0b01100110
  - Bit shift right: 0b00110011 >> 1 == 0b00011001
- This is essentially multiplication or division by powers of 2, respectively, since shifting means changing places, identical to how we move a decimal point around when we either multiply or divide by 10.
  - 0s fill in the spaces left behind when shifting in either direction
- Used mostly for hardware level communication alongside the other Boolean Algebra operations.

Bit Shifting Quirks and Precedence

- Note that the symbols for bit shifting in C/C++ are the same as the ones used for cout and cin.
- << and >> have higher precedence than bitwise logic.
- If you try to do bitwise logic in the same lines as cout and cin, then compiler errors may occur if parentheses are not included in the right places.
  - cout << 0b1010 ^ 0b0101 << endl;
    - This will not compile. The program will try to shift 0b0101 by endl, which makes no sense and thus returns an error.
  - cout << (0b1010 ^ 0b0101) << endl;
    - This will compile. The parentheses forces the bitwise operations to carry out first, and then the output can be properly handled.

Utilizing Bit Shifts

Program

```
int x = 12;
int main()
{
        cout << x << 2 << endl;
        // Without parentheses, the above is just 12 and 2

        // Proper syntax is as follows:
        cout << (x << 2) << endl;
        cout << (x >> 2) << endl;
        // Bit shift by two places, int supports 32 bits
        // The top line is essentially multiply by 2^2 = 4
        // The bottom line is essentially divide by 2^2 = 4
        return 0;
}
```

Console

```
➢ ./a.exe
122
48
3
```