CSM 61B
Spring 2018

Bits
March 22, 2018

1 Material

Recall the following bit operations:

1. Bitwise OR |:
   Compares corresponding bits of two operands. Returns 1 if either is 1, 0 otherwise.
2. Bitwise AND &:
   Compares corresponding bits of two operands. Returns 1 if both bits are 1, 0 otherwise.
3. Bitwise Complement $\sim$:
   Inverts the bit pattern. Returns 1 if the bit is 0, 0 if the bit is 1.
4. Bitwise XOR $\wedge$:
   Compares corresponding bits of two operands. Returns 1 if corresponding bits are different, 0 otherwise.
5. Signed Left Shift <<:
   Shifts a bit pattern to the left by certain number of bits, filling in gaps with the sign bit (1 if negative, 0 if positive).
6. Signed Right Shift >>:
   Shifts a bit pattern to the right by certain number of bits, filling in gaps with the sign bit (1 if negative, 0 if positive).
7. Unsigned Right Shift >>>:
   Shifts a bit pattern to the right by certain number of bits, filling in gaps with the zero bit.

Bit Operations				
a	b	$a \mid b$ (OR)	$a \& b$ (AND)	$a \wedge b$ (XOR)
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

2 Problems

2.1 Return the n^{th} bit of x.
Assume $0 \leq n \leq 31$

```
public int getBit(int x, int n) {
    return (x  & (1 << n)) >> n;
}
```

2.2 Return x with the n^{th} bit of the value of x set to v.
Assume $0 \leq n \leq 31$, and v is 0 or 1.

```
public int setBit(int x, int n, int v) {
    return v ? x | (1 << n) : x & ~(1 << n);
}
```

```
public int setBit(int x, int n, int v) {
if (v == 1) {
     return x | (1 << n)
    } else {
     return x & ~(1<<n)
    }
    return (x & (-1 - (1<<n))) + (v<<n);
}
```

2.3 Return x with the n^{th} bit of the value of x flipped.
Assume $0 \leq n \leq 31$

```
public int flipBit(int  x, int n) {
    return x ^ (1 << n)
}
```

2.4 Write a method to test whether or not a positive integer is a power of 2. Your method should run in $O(1)$ time and use $O(1)$ space.

```
if (((num & (num-1)) == 0) || (num == 0))
    return true;
else
    return false;
```