THE UNIVERSITY of NEW ORLEANS

DEPARTMENT OF COMPUTER SCIENCE

CSCI 2467, Fall 2017

Class Activity: Two's complement, bitwise and logical operators

Tuesday, September 5

1 Response Sheet Instructions

Please use these pages to record your group's answers. You can use the question pages as a workspace, then use this page for your final answers.

Names
Answer
Key

2 Review of Negative Integers

1. The leftmost bit of a *non-negative* number is always:

   0

   The leftmost bit of a *negative* number is always:

   1

2. Fill in the table below.

All possible 3-bit values

000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

Bits	Most pos.	Most neg.
1	0	-1
2	1	-2
3	3	-4
4	7	-8

3. Write an expression (in terms of N) that gives the *most positive* number that can be represented by a N-bit two's complement number.

   $2^{N-1} - 1$

4. Write an expression (in terms of N) that gives the *most negative* number that can be represented by a N-bit two's complement number.

   -2^{N-1}

3 Bit-level Operations

7. Fill in the table with your answers. Note the arrows showing the order of conversions in the table.

Hex.	Binary
~(0xF0F0) →	~(1111 0000 1111 0000)
	↓
0x0F0F ←	0000 1111 0000 1111
0x3C3C	0011 1100 0011 1100
0xC3C3	1100 0011 1100 0011
0xCAFE	1100 1010 1111 1110
0x3501	0011 0101 0000 0001
0x0000	0000 0000 0000 0000
0xFFFF	1111 1111 1111 1111

8. Fill in the table using bitwise AND (&):

Dec	Bin	X & 0x1
-2	1110	0000
-1	1111	0001
0	0000	0000
1	0001	0001
2	0010	0000

9. For which numbers was the value ANDed with 0x1 *not* 0000? What is a comon property of these integers?

-1 and 1 → both are odd numbers (last bit is a 1)

10. Explain the expression `(X & FLAG) == FLAG`. When does it return true (1)? When does it return false (0)?

Returns true → when any bit in FLAG is set in X

Returns false → when NO bits set in FLAG are set in X

11. How is OR (|) used in the open() example?

OR (|) sets the relevant bits for each File access mode, combining those bits into one value (int) to be passed to open()

12. What constant value should you use as a bitmask in order to mask out all bits except the sign bit?

```
int signbit = x & 0x80000000 ;
```

13. Provide an equivalent bitmask, but use a small constant and a shift instead of a large constant:

```
int signbit = x & 1<<31 ;
```

14. What is the value of `signbit` if `x` is positive? Negative?

If x is positive, signbit == 0

If x is negative signbit == 0x80000000 → which is the same as TMin

15. Use a shift and a bitmask to convert `signbit` to either 0 (for positive) or 1 (for negative):

or just 1 ↓

`int isNeg = ( signbit >> 31 ) & 0x1;` `int isNeg = ( x >> 31 ) & 0x1;`

Could you also write this expression in terms of x?

4 Logical operations

1. With 4-bit values, how many values are false?

   1 (zero is the only one)

   How many are true?

   15 (all non-zero values)

Note: 4-bit numbers have 16 possible values

2. Evaluate the following expression:
   `(0x3 && 0xC) == (0x3 & 0xC)`

[&& is logical AND] both 0x3 and 0xC are non-zero so they are both true, so this logical AND evaluates to 1

[& is bitwise AND] 0x3 → 0011, 0xC → 1100, 0011 & 1100 = 0000 = 0

Is this result what you expected? Why or why not?

Expression simplifies to 1 == 0. Those two values are not equal, so that expression evaluates to false represented by 0

Even though the two sides look almost identical, they are not equal. This could be surprising, until we know the difference between & and &&

3. Test whether $!!X == X$ holds across different values of X.

X	!X	!!X
-1	0	1
0	1	0
1	0	1
2	0	1

4. Using ~ instead of !:

X	~X	~~X
-1	0	-1
0	-1	0
1	-2	1
2	-3	2

How does this differ from the previous result? Why?

With ! there are only two possible results, 0 and 1.
~ has many possible results, in fact every unique int has a unique complement.
Also: ~(~(x)) == x is true (not for !)