CSC 209 Review 9 Solution

August 31, 2020

1. a) 0

Notes

- `a)` is 0 because (`i >> 1 + j >> 1 = i >> 10 >> 1 = 0`)
- **Bitwise Shift Operators**
  - has lower precedence than arithematic operators

    Example:

    `i << 2 + 1` means `i << (2+1)` and not `(i << 2) + 1`

  - << : Left Shift
  - >> : Right Shift
  - *Tip:* Always shift only on unsigned numbers for portability

    Example

```
unsigned short i, j;

i = 13;       /* i is now 13 (binary 0000000000001101) */
j = i << 2;   /* j is now 52 (binary 0000000000110100) */
j = i >> 2;   /* j is now  3 (binary 0000000000000011) */
```

Shifts to left

Shifts to right

As these examples show, neither operator modifies its operands. To modify a variable by shifting its bits, we'd use the compound assignment operators <<= and >>=:

```
i = 13;       /* i is now 13 (binary 0000000000001101) */
i <<= 2;      /* i is now 52 (binary 0000000000110100) */
i >>= 2;      /* i is now 13 (binary 0000000000001101) */
```

  - >>= / <<= : Are bitwise shift equivalent of + =

b) 0

Notes

- `i` is `1111111111111111`
- `i` is `0000000000000000`
- so `i & i = 0`
- : Bitwise complement (NOT)

a	∼ a
0	1
1	0

Example:

```
    0   1   1   1   //<- this is 7
    --------------
    1   0   0   0   //<- this is 8

    so, ~ 7 = 8
```

- `&`: Bitwise *and*

a	b	a & b
0	0	0
0	1	1
1	0	0
1	1	1

Example:

```
    0   1   1   1   //<- this is 7
    0   1   0   0   //<- this is 4
    --------------
    0   1   0   0   //<- this is 4

    so, 7 & 4 = 4
```

- : Bitwise *exclusive or*
- |: Bitwise *inclusive or*

c) 0

Notes

- ^: Bitwise XOR

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

Example:

```
    0   1   1   1   //<- this is 7
    0   1   0   0   //<- this is 4
    --------------
    0   0   1   1   //<- this is 3

    so, 7 ^ 4 = 3

```