

# CSC 209 Review 9 Solution

August 31, 2020

1. a) 0

## Notes

- a) is 0 because  $(i \gg 1 + j \gg 1 = i \gg 10 \gg 1 = 0)$
- **Bitwise Shift Operators**
  - has lower precedence than arithmetic operators

## Example:

$i \ll 2 + 1$  means  $i \ll (2+1)$  and not  $(i \ll 2) + 1$

- $\ll$  : Left Shift
- $\gg$  : 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) */
```

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

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

Shifts to left

Shifts to right

- $\gg=$  /  $\ll=$  : Are bitwise shift equivalent of  $+=$

b) 0

## Notes

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

a	$\sim a$
0	1
1	0

**Example:**

```

1      0   1   1   1   //<- this is 7
2      -----
3      1   0   0   0   //<- this is 8
4
5      so, ~ 7 = 8

```

- `&`: Bitwise *and*

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

**Example:**

```

1      0   1   1   1   //<- this is 7
2      0   1   0   0   //<- this is 4
3      -----
4      0   1   0   0   //<- this is 4
5
6      so, 7 & 4 = 4

```

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

c) 1

Notes

- `i` is 1111111111111110
- `j` is 0000000000000000
- `i & j` is 0000000000000000 or 1
- `i & j ^ k` is 1

- $\wedge$ : Bitwise XOR

a	b	$a \wedge b$
0	0	0
0	1	1
1	0	1
1	1	0

### Example:

```

1      0   1   1   1   //<- this is 7
2      0   1   0   0   //<- this is 4
3      -----
4      0   0   1   1   //<- this is 3
5
6      so, 7 ^ 4 = 3
7

```

d) 0

### Example

- i is 0000000000000111
- j is 0000000000001000
- $i \wedge j$  is 0000000000000000 or 0
- k is 0000000000001001
- $i \wedge j \& k$  is 0000000000000000 or 0

### Correct Solution

15

### Notes

- There is a precedence to the order of operations

Highest:	$\sim$
	$\&$
	$\wedge$
Lowest:	$ $