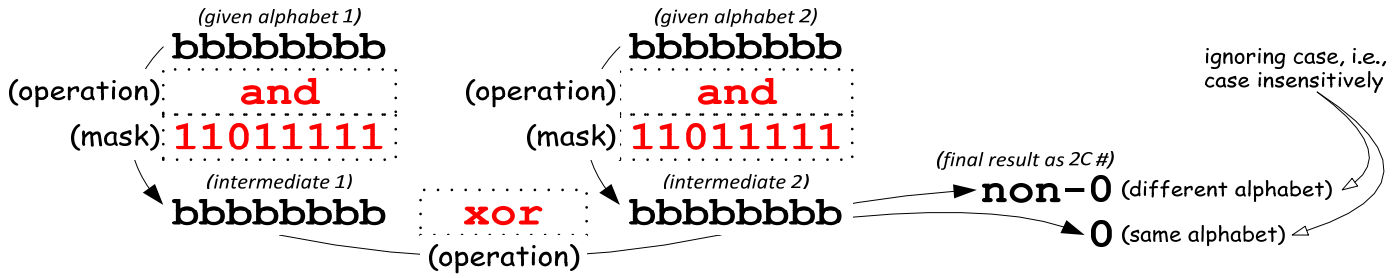
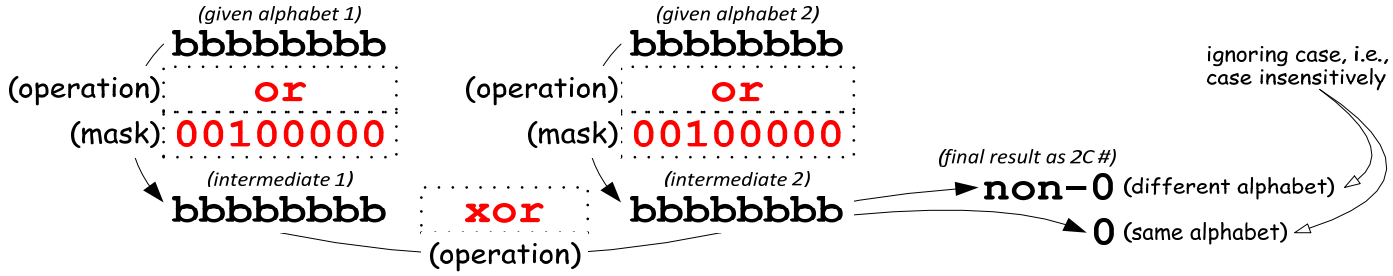
Solutions: Drills and Challenges on *Bitwise Operations and Masking*

- For (1):
  AND **given** with **0000000000010000** (as `mask1`) to get **intermediate1**.
  SHIFT RIGHT LOGICAL **intermediate1** by 4 positions.
  AND **given** with **0000000100000000** (as `mask2`) to get **intermediate2**.
  SHIFT RIGHT LOGICAL **intermediate2** by 8 positions.
  AND **intermediate1** with **intermediate2** to get **desired**.
  For (2):
  XOR **desired** (obtained in (1)) with **0000000000000001** to obtain a new **desired**.

- One solution ("***force uppercase***"):

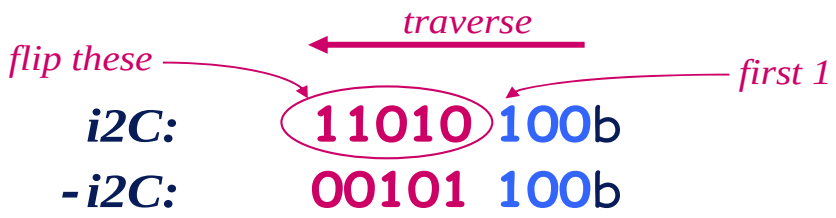


  Another solution ("***force lowercase***"):



- First ***shift*** `num` ***right*** (***logical***ly) by 16 positions (note that 65536 = 2^{16}) to get `quot` = `num/65536`.
  Then ***and*** `quot` with 255 (= 256 - 1, which is `0...011111111` in binary) to get the desired.

- Putting the "in-1-step" way of finding 2's complement, namely



  to use on some positive 2C number (our interest is in some power of 2, so don't have to worry about negatives), we see that the ***result of ANDing*** `i2C` and `-i2C` must be one of the following (single 1 appearing in the midst of 0's):

```
0........01
......
0...010...0
......
010.......0
```

  This result can be identical to `i2C` only if `i2C`'s representation contains only a single 1 (which means `i2C` is a power of 2).
  The above doesn't apply to **0** (which is the special case).