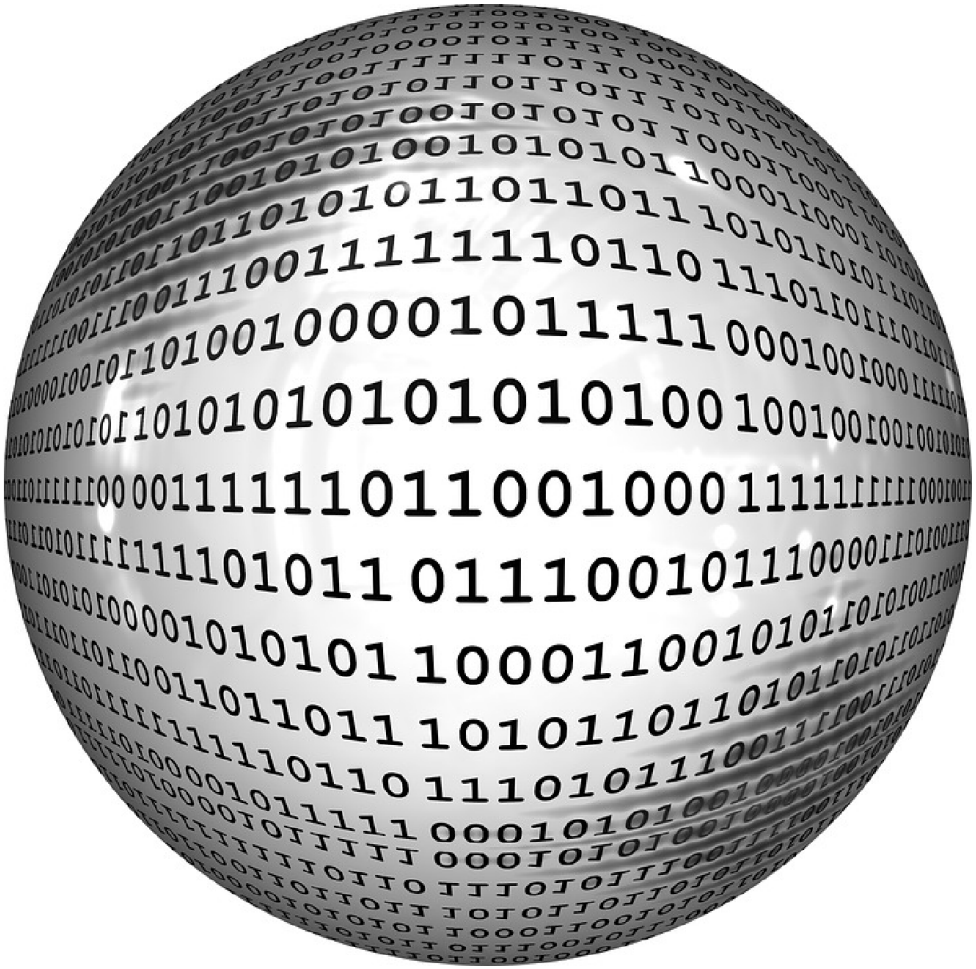
Bitwise operators



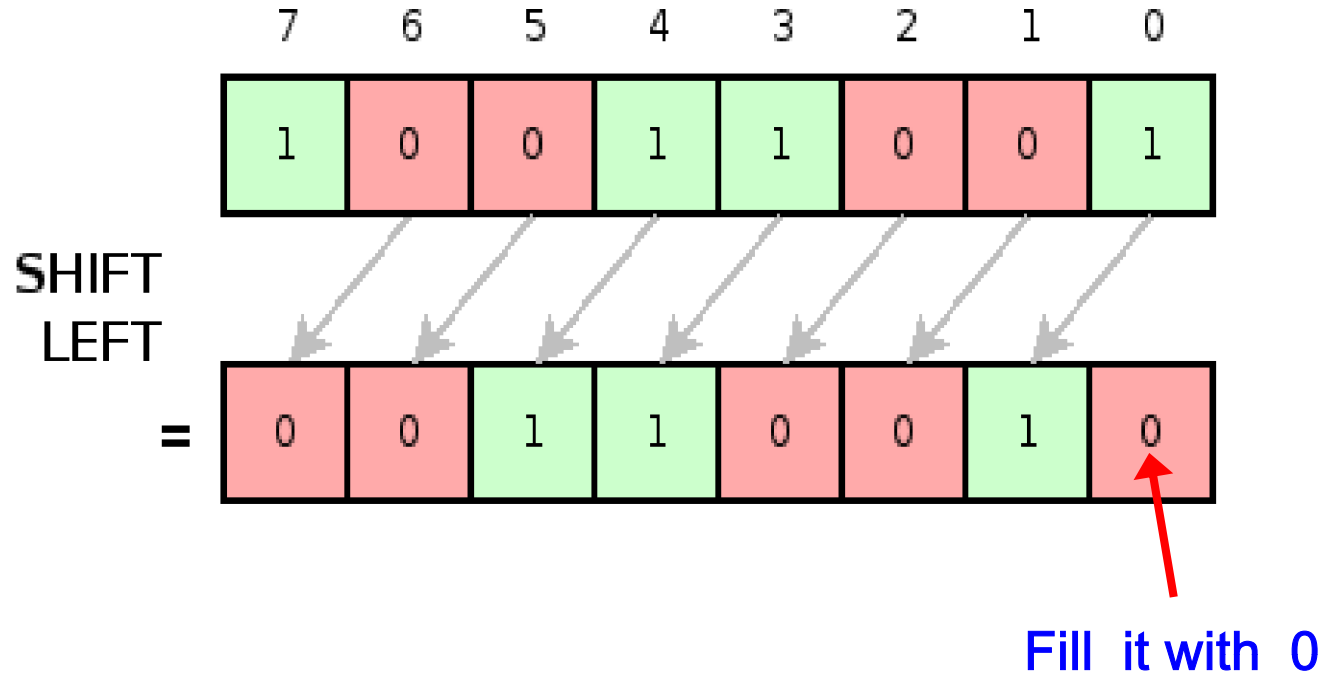
Reading list

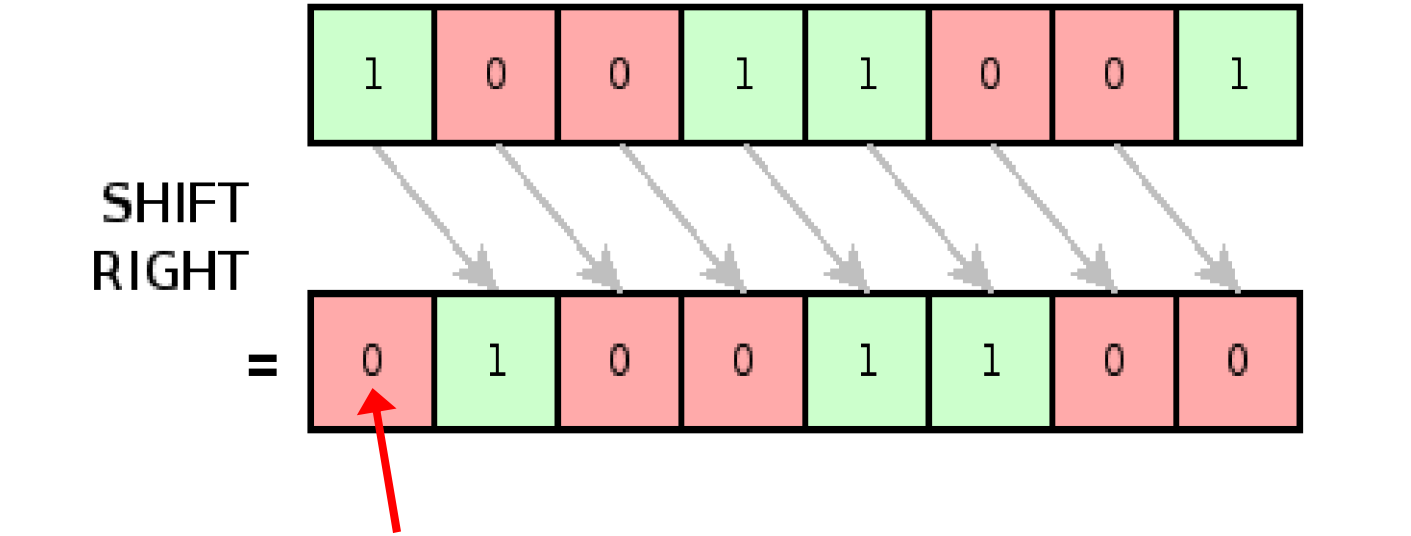
- Read Kernighan & Ritchie Page 48-49
- Lecture Slides
- Reading material from the course website

A2+B2: Lab exam (online) after midterm break

Most Commonly Used Laws of Operators

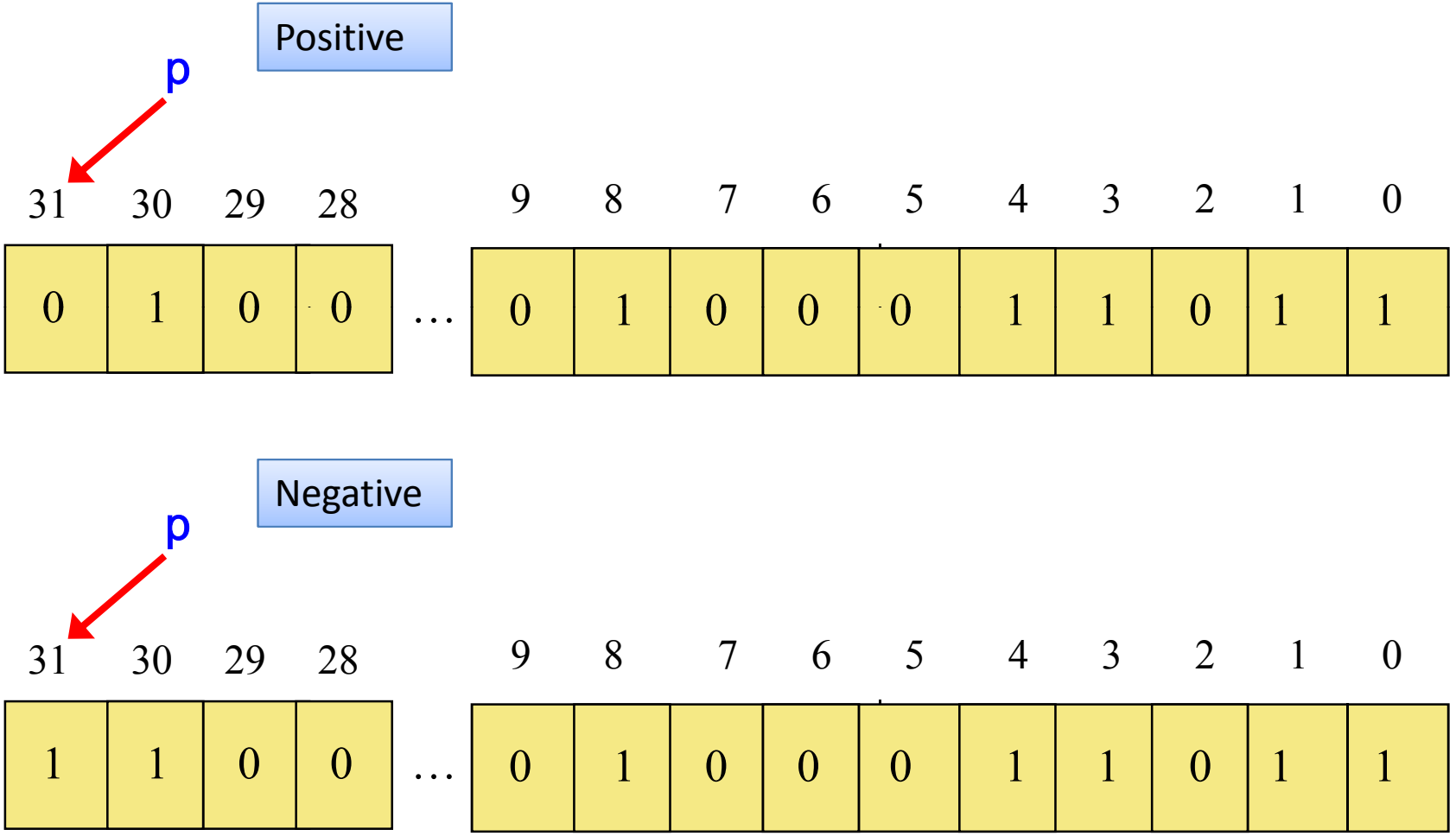
a & 0 → 0	a | 0 → a	a ^ 0 → a
a & 1 → a	a | 1 → 1	a ^ 1 → ~a
a & a → a	a | a → a	a ^ a → 0


7 6 5 4 3 2 1 0
1 0 0 1 1 0 0 1
SHIFT
LEFT
=
0 0 1 1 0 0 1 0
Fill it with 0


1 0 0 1 1 0 0 1
SHIFT
RIGHT
=
0 1 0 0 1 1 0 0
For signed numbers fill it with 1

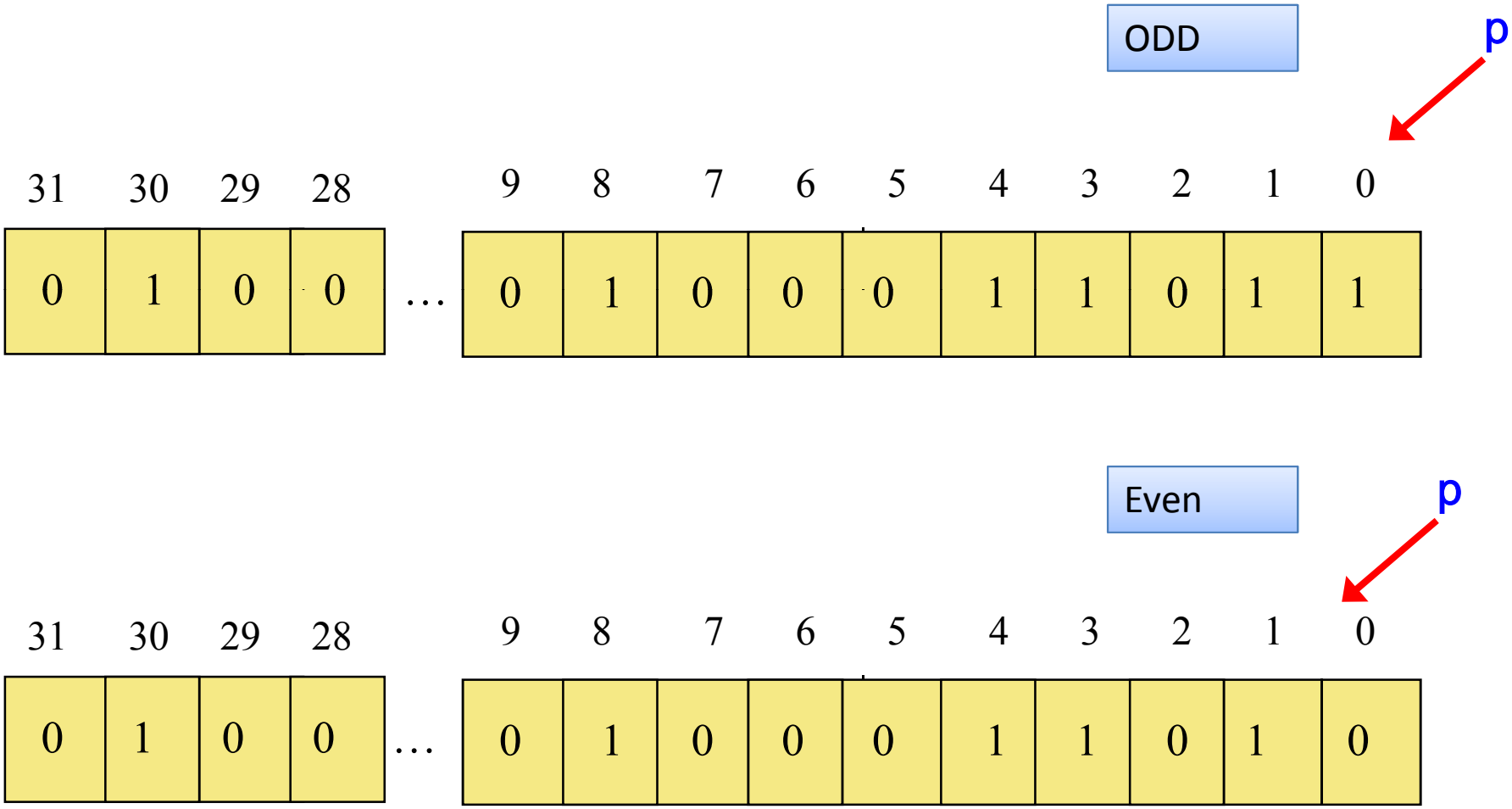
Some quick examples

Determining a number positive or negative?



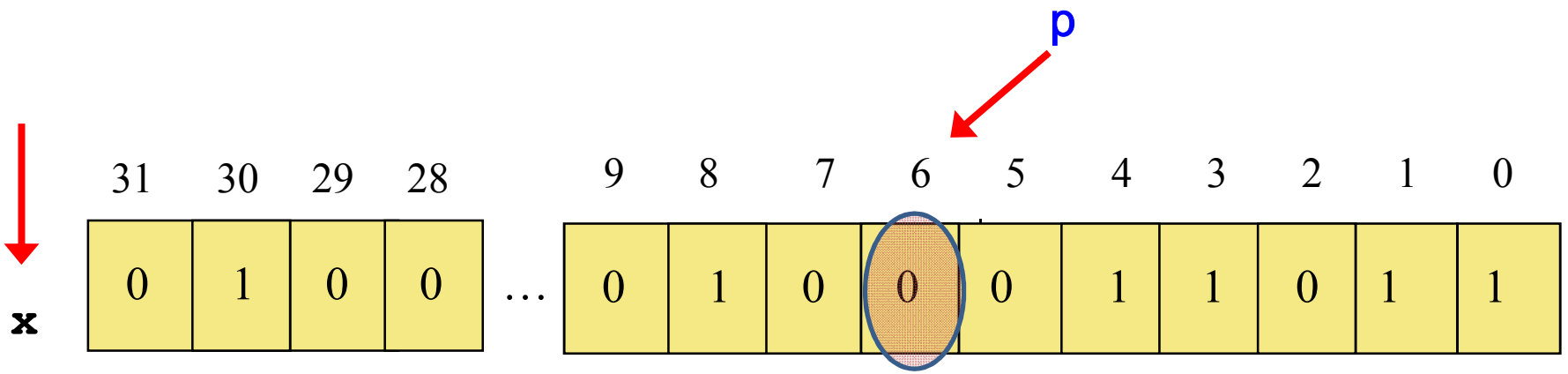
Some quick examples

Determining a number odd or even?



setBit(x,p)

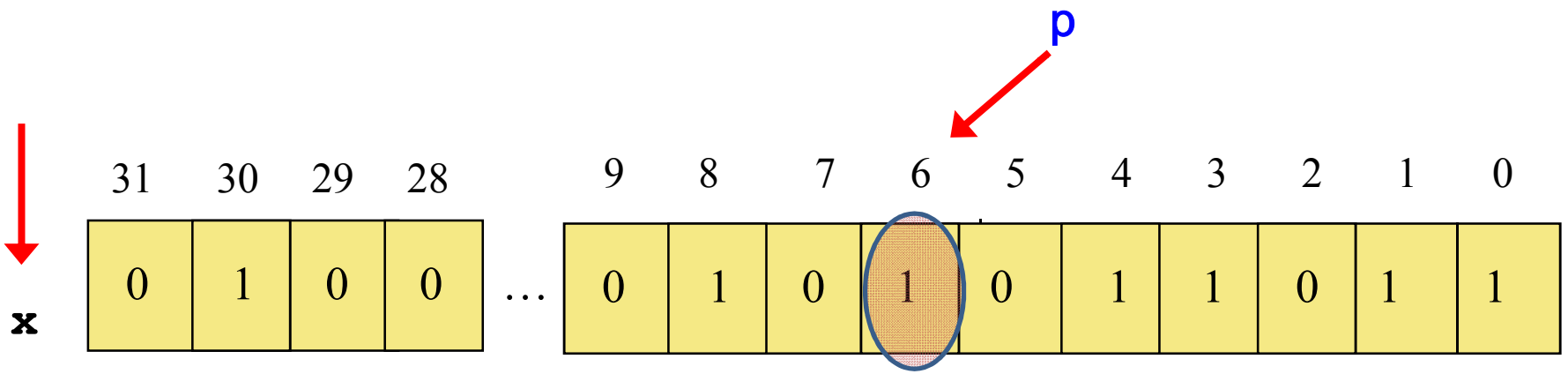
Write down a function setBit(x,p) that will **set** a bit of integer x at position p leaving other bits unchanged. Assume 0 ≤ p ≤31



setBit(x, 6)

setBit(x,p)

Write down a function setBit(x,p) that will **set** a bit of integer x at position p leaving other bits unchanged. Assume $0 \leq p \leq 31$

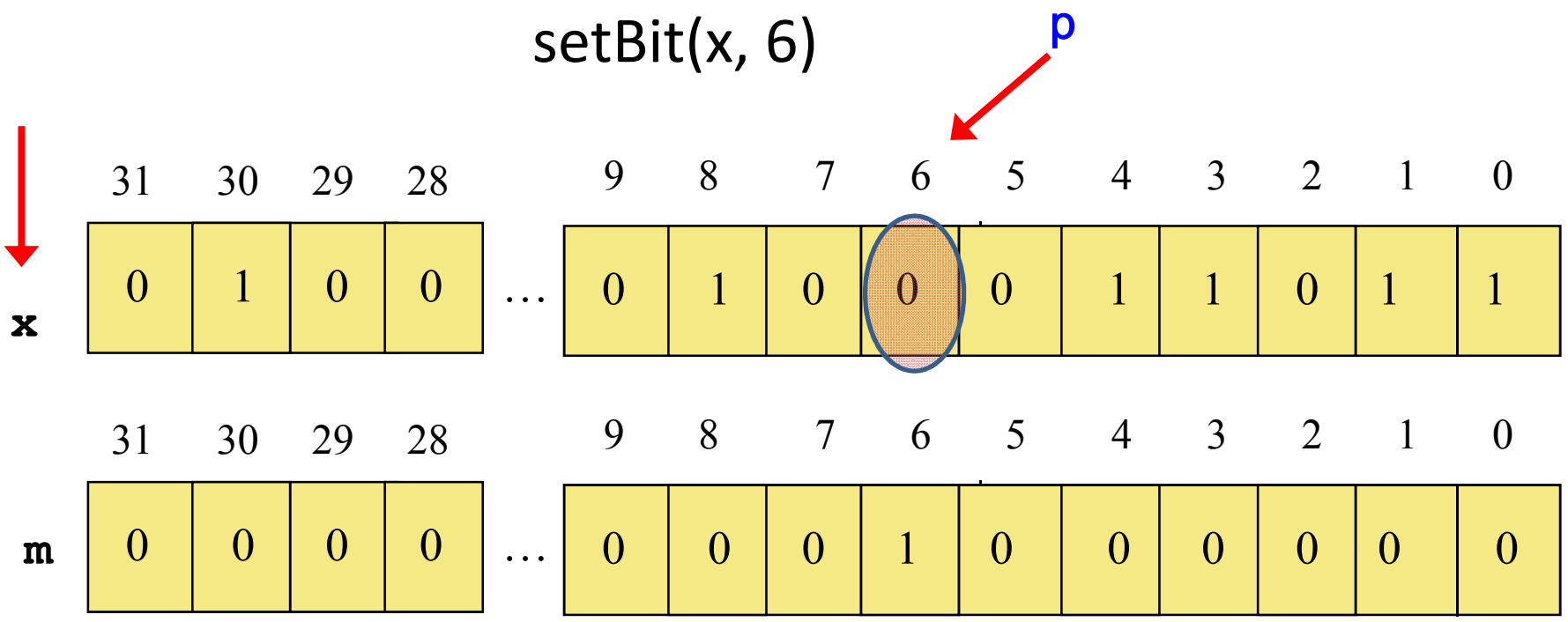


setBit(x, 6)

a | 0 ➔ a
a | 1 ➔ 1
a | a ➔ a

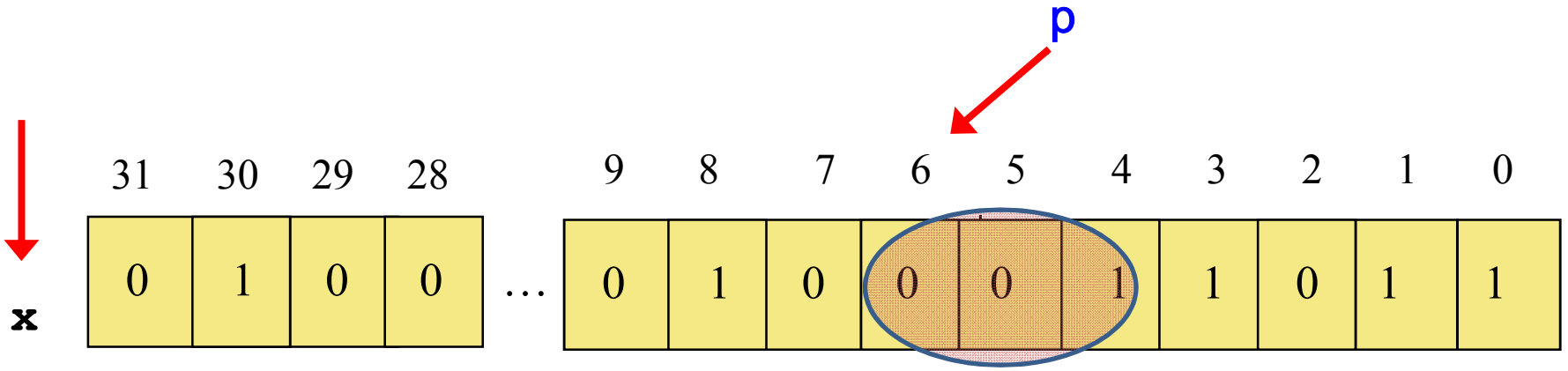
setBit(x,p)

Write down a function setBit(x,p) that will **set** a bit of integer x at position p leaving other bits unchanged. Assume 0 ≤ p ≤31



setBits(x,p,n)

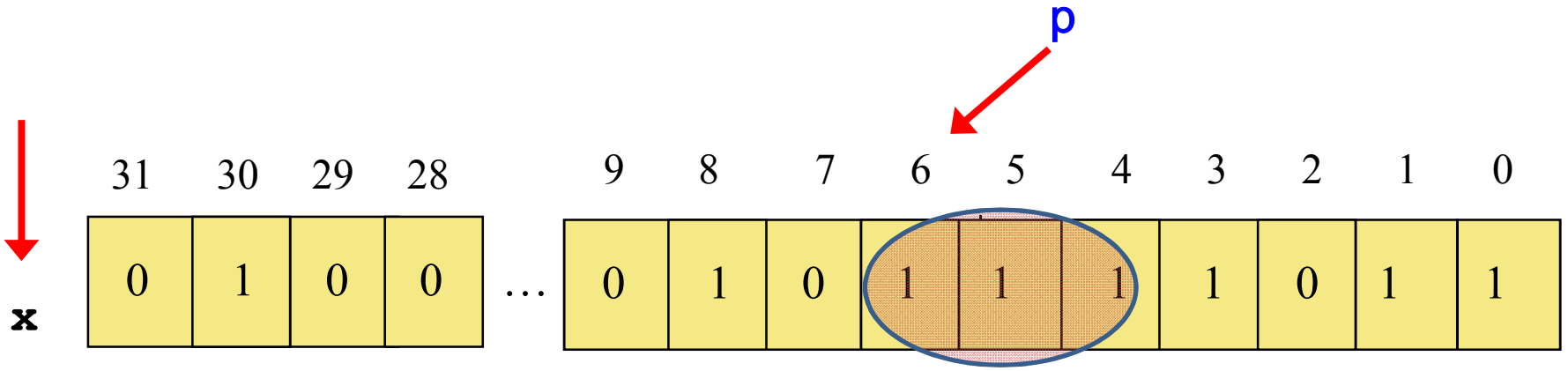
Write down a function setBits(x,p,n) that will **set** the n bits of the integer x starting from position p leaving other bits unchanged. Assume 0 ≤ p ≤31 and n ≤ p+1



setBits(x, 6, 3)

setBits(x,p,n)

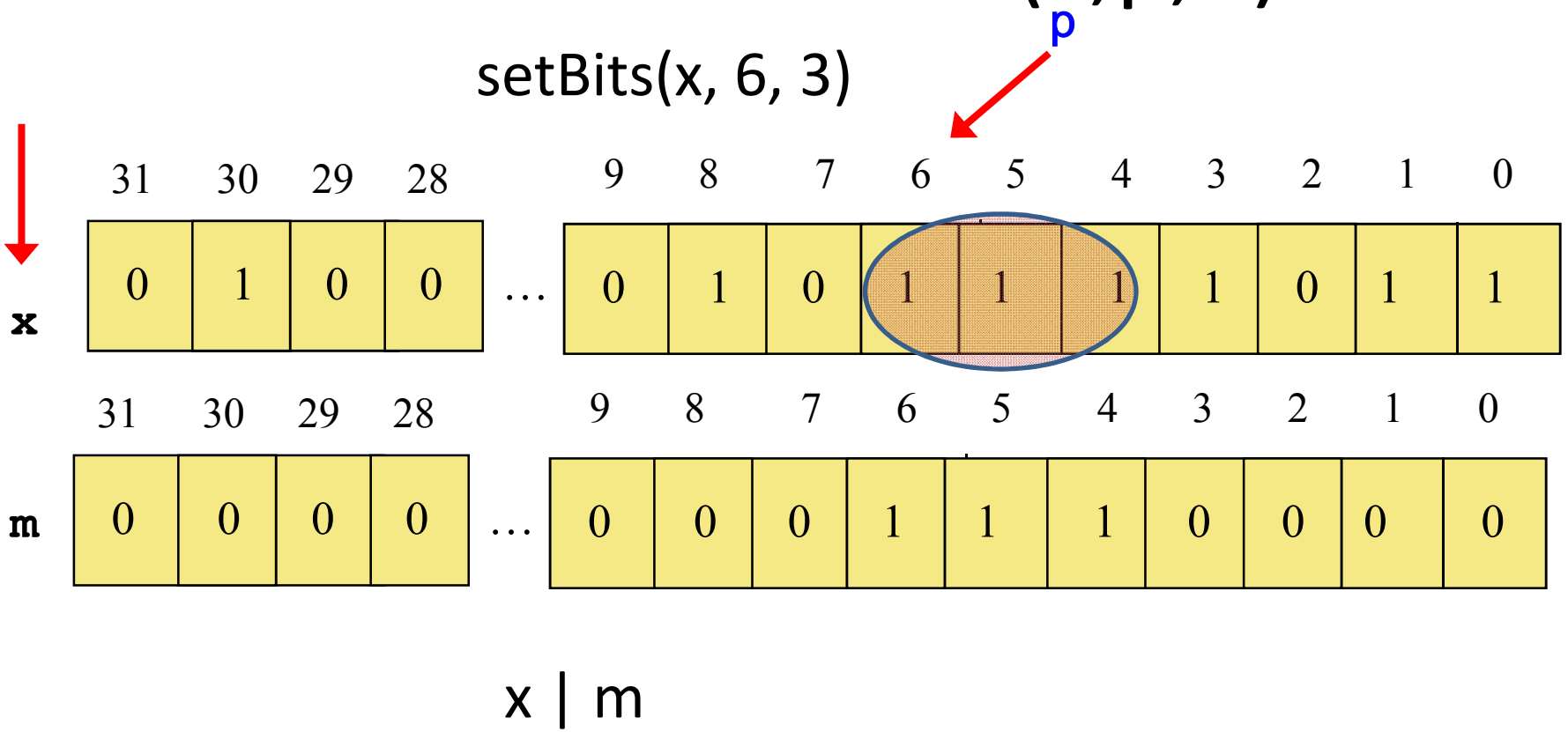
Write down a function setBits(x,p,n) that will **set** the n bits of the integer x starting from position p leaving other bits unchanged. Assume 0 ≤ p ≤31 and n ≤ p+1



setBits(x, 6, 3)

Call setBit(x,p) in a loop

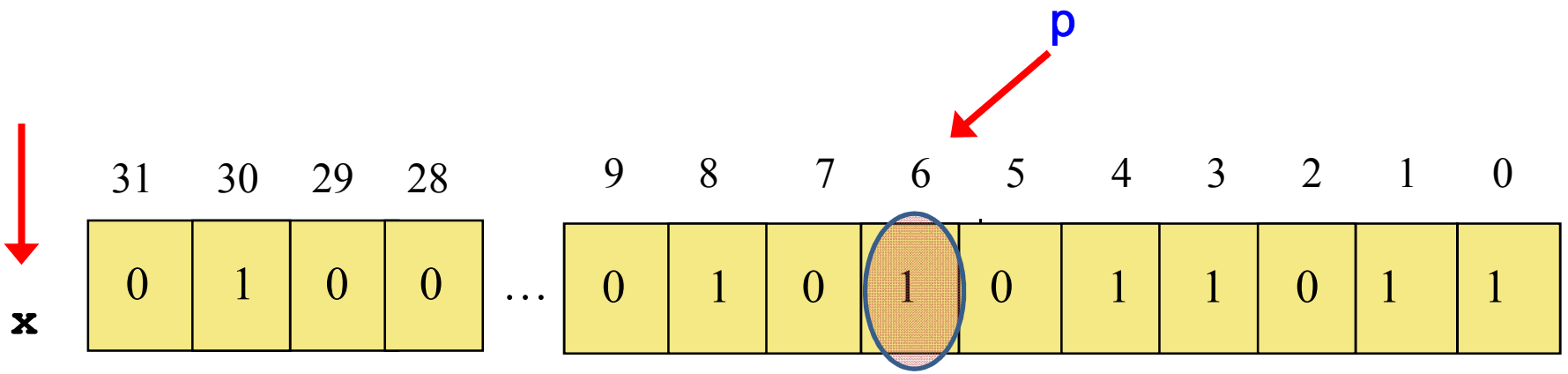
Efficient setBits(x,p,n)



x | m

resetBit(x,p)

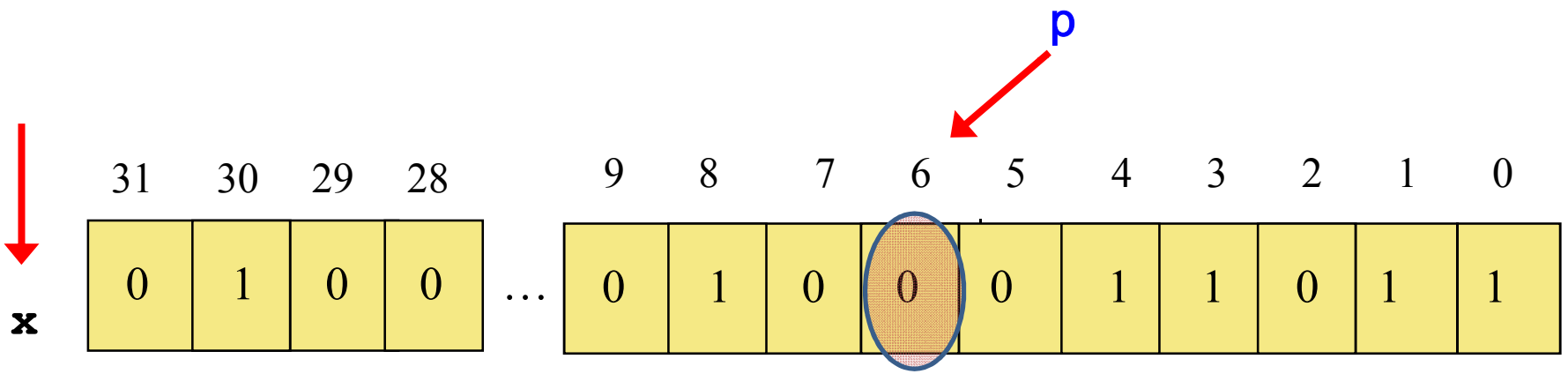
Write down a function resetBit(x,p) that will **reset** a bit of integer x at position p leaving other bits unchanged. Assume 0 ≤ p ≤31



resetBit(x, 6)

resetBit(x,p)

Write down a function resetBit(x,p) that will **reset** a bit of integer x at position p leaving other bits unchanged. Assume 0 ≤ p ≤31

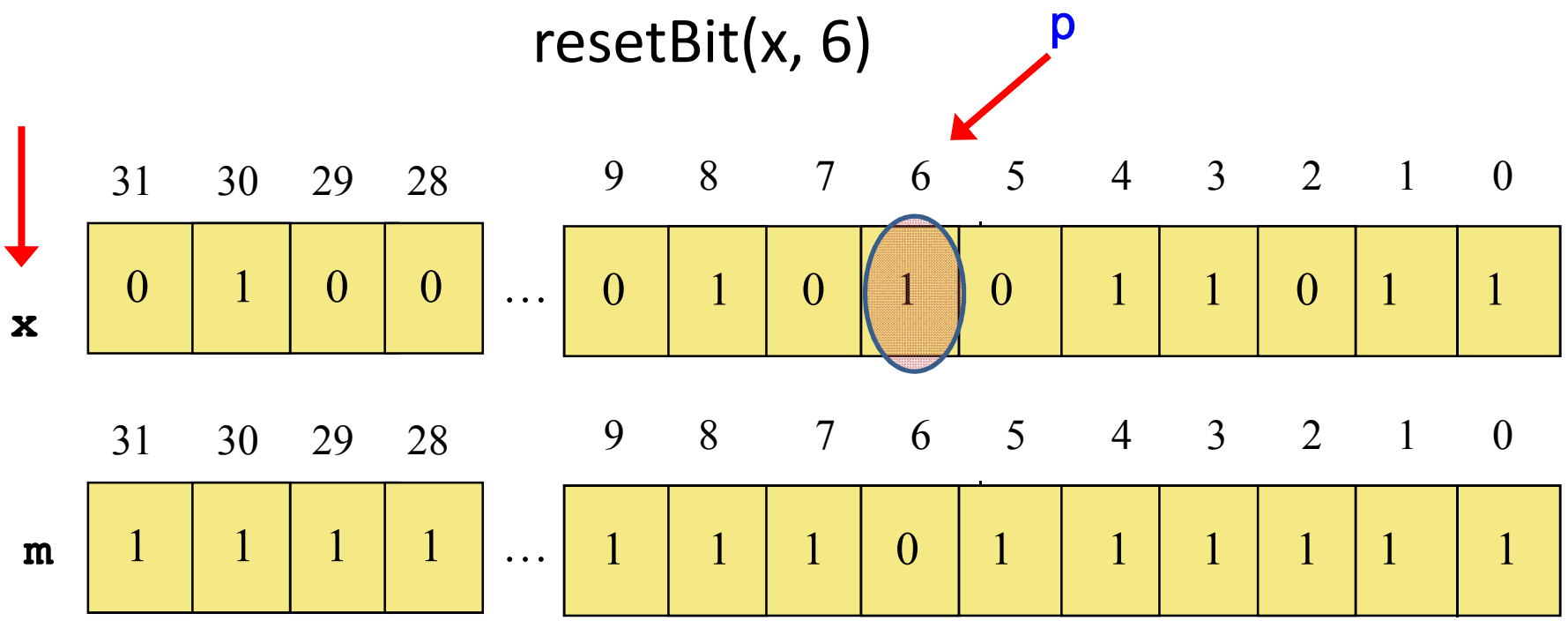


resetBit(x, 6)

a & 0 → 0
a & 1 → a
a & a → a

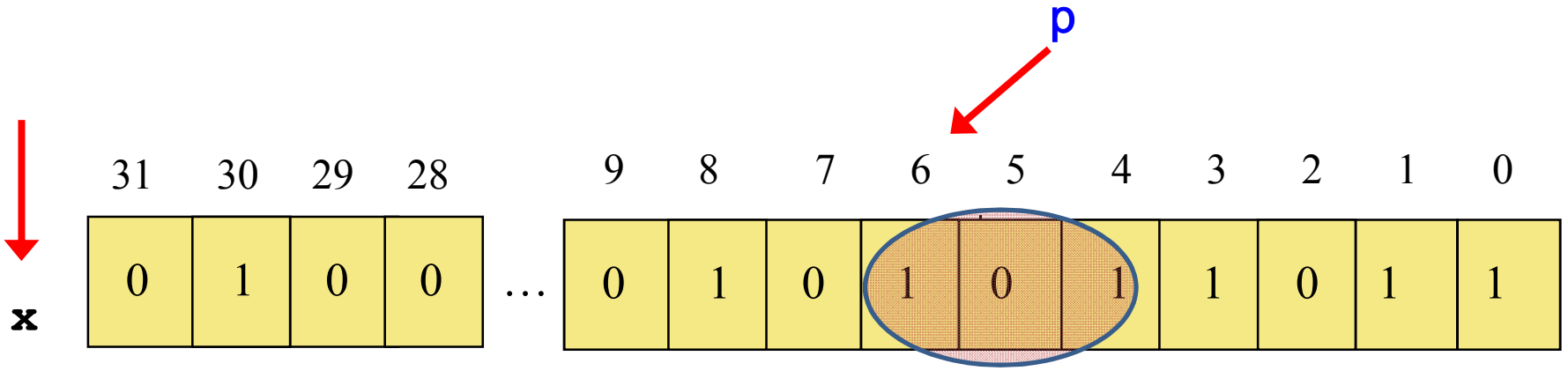
resetBit(x,p)

Write down a function resetBit(x,p) that will **reset** a bit of integer x at position p leaving other bits unchanged. Assume 0 ≤ p ≤31



resetBits(x,p,n)

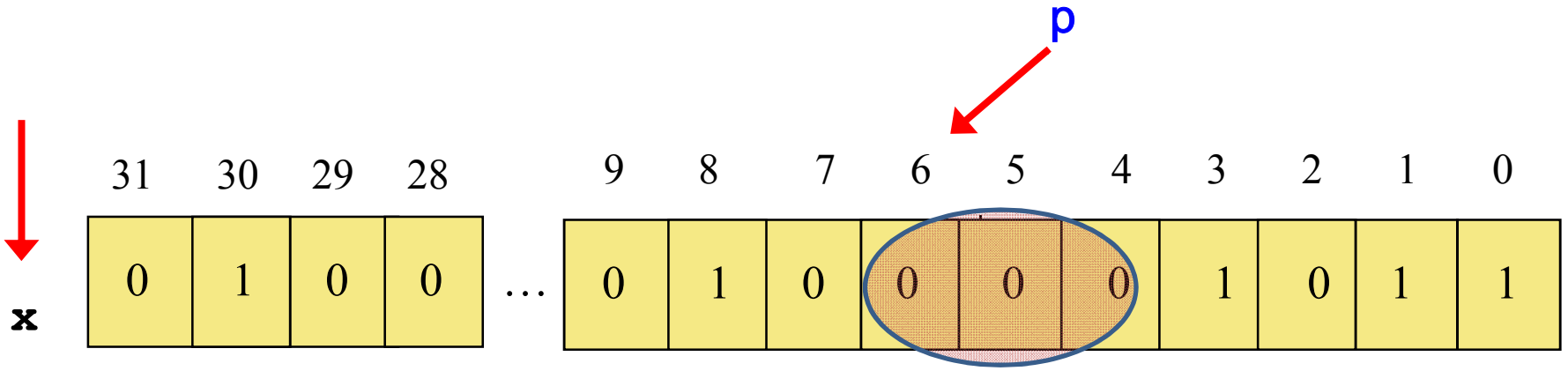
Write down a function resetBits(x,p,n) that will **reset** the n bits of the integer x starting from position p leaving other bits unchanged. Assume 0 ≤ p ≤31 and n ≤ p+1



resetBits(x, 6, 3)

resetBits(x,p,n)

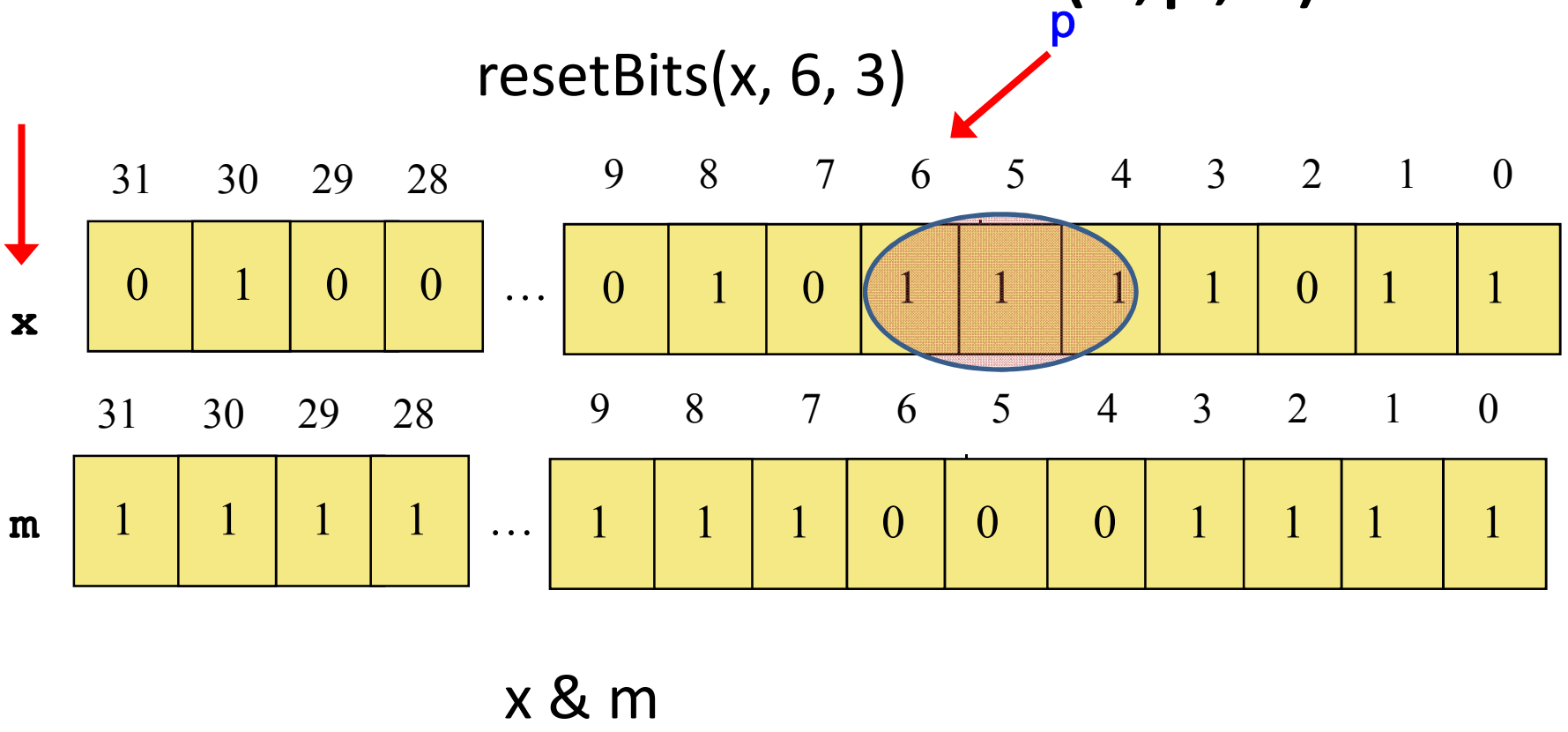
Write down a function resetBits(x,p,n) that will **reset** the n bits of the integer x starting from position p leaving other bits unchanged. Assume $0 \le p \le 31$ and $n \le p+1$



resetBits(x, 6, 3)

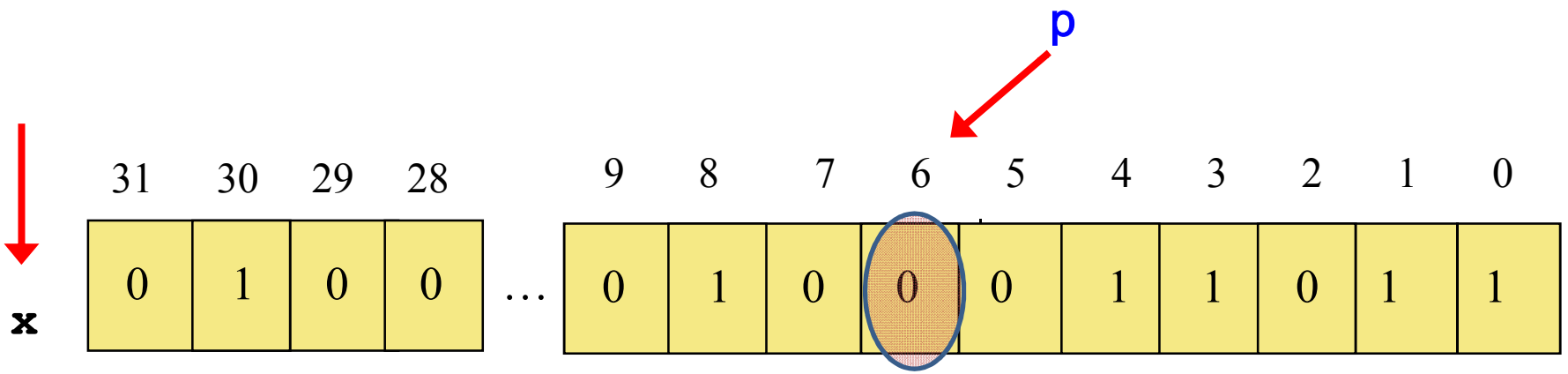
Call resetBit(x,p) in a loop

Efficient resetBits(x,p,n)



invertBit(x,p)

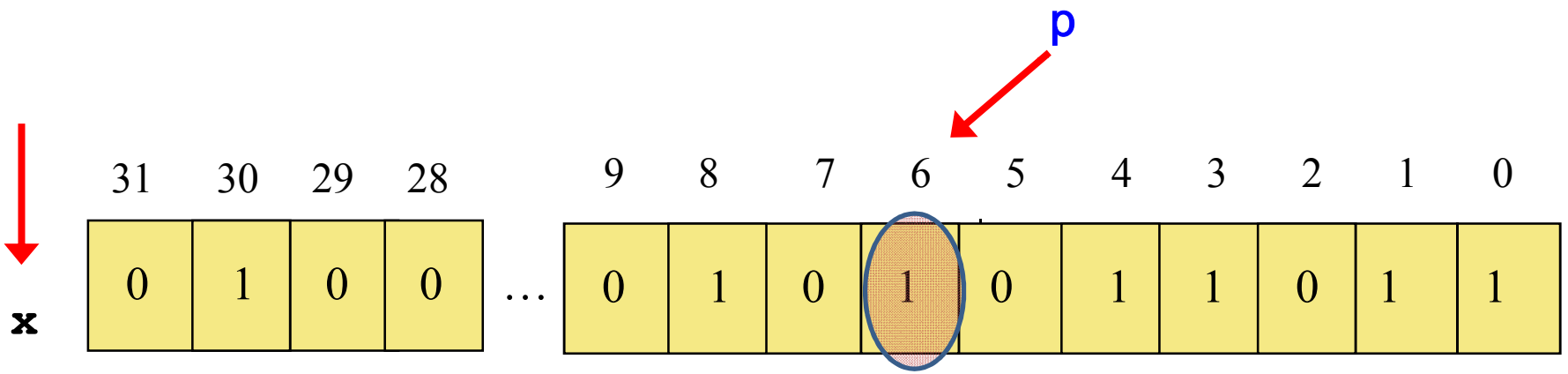
Write down a function invertBit(x,p) that will **invert** a bit of integer x at position p leaving other bits unchanged. Assume 0 ≤ p ≤31



invertBit(x, 6)

invertBit(x,p)

Write down a function setBit(x,p) that will **invert** a bit of integer x at position p leaving other bits unchanged. Assume 0 ≤ p ≤31

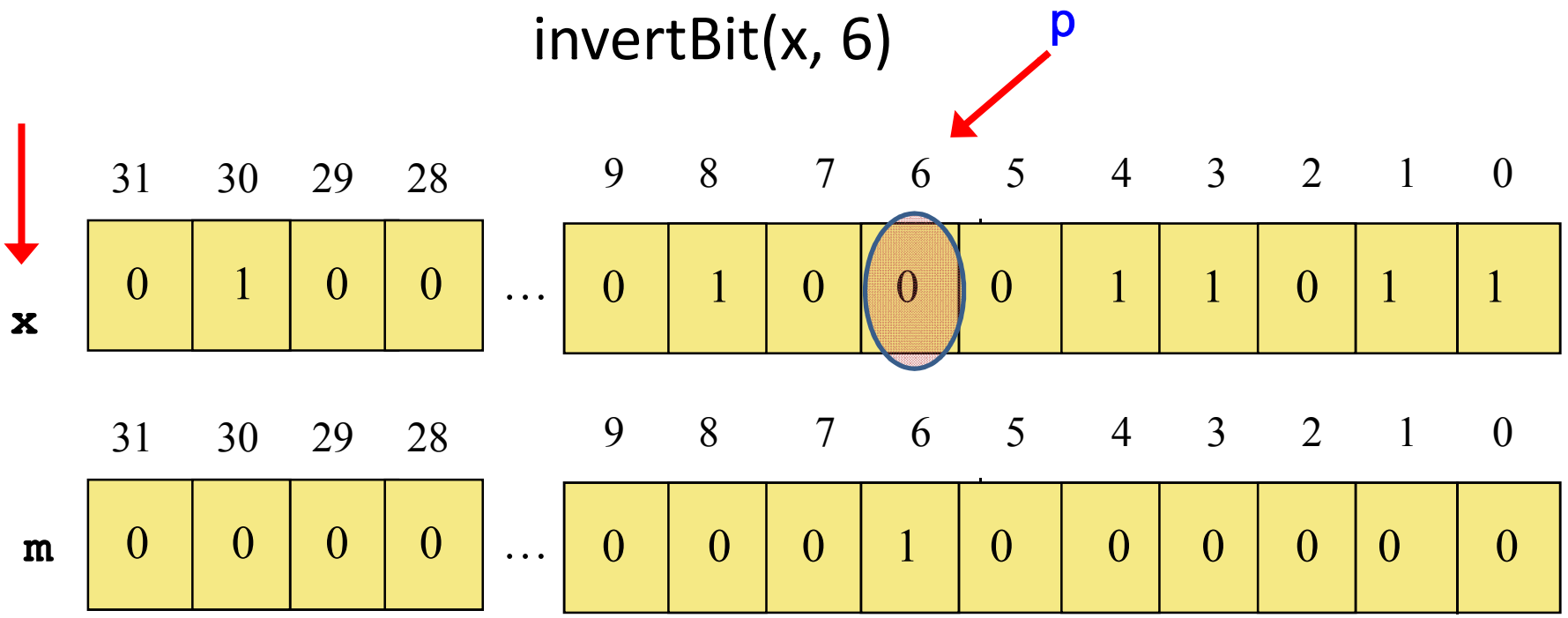


invertBit(x, 6)

a ^ 0 ➔ a
a ^ 1 ➔ ~a
a ^ a ➔ 0

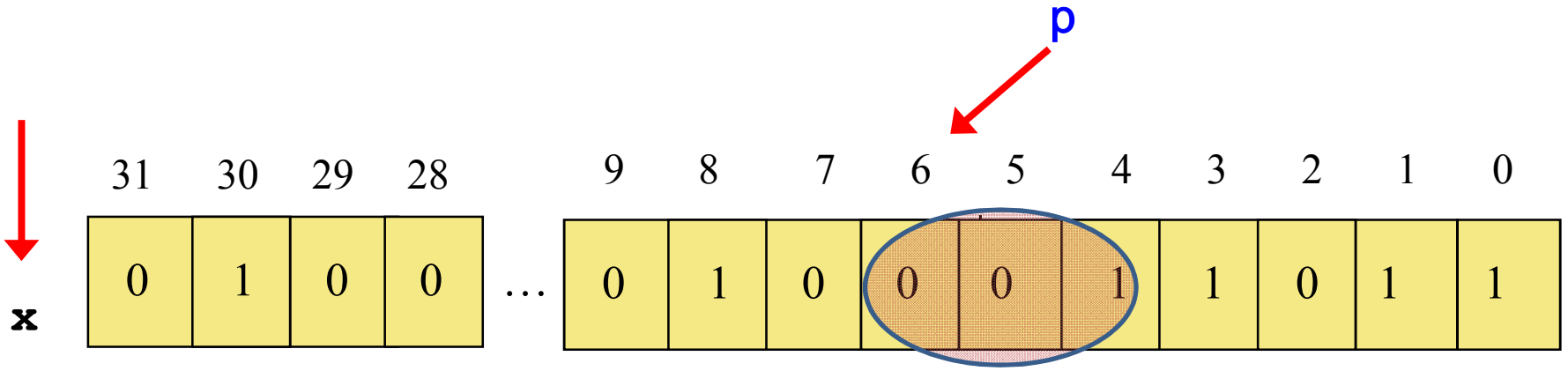
invertBit(x,p)

Write down a function invertBit(x,p) that will **invert** a bit of integer x at position p leaving other bits unchanged. Assume 0 ≤ p ≤31



invertBits(x,p,n)

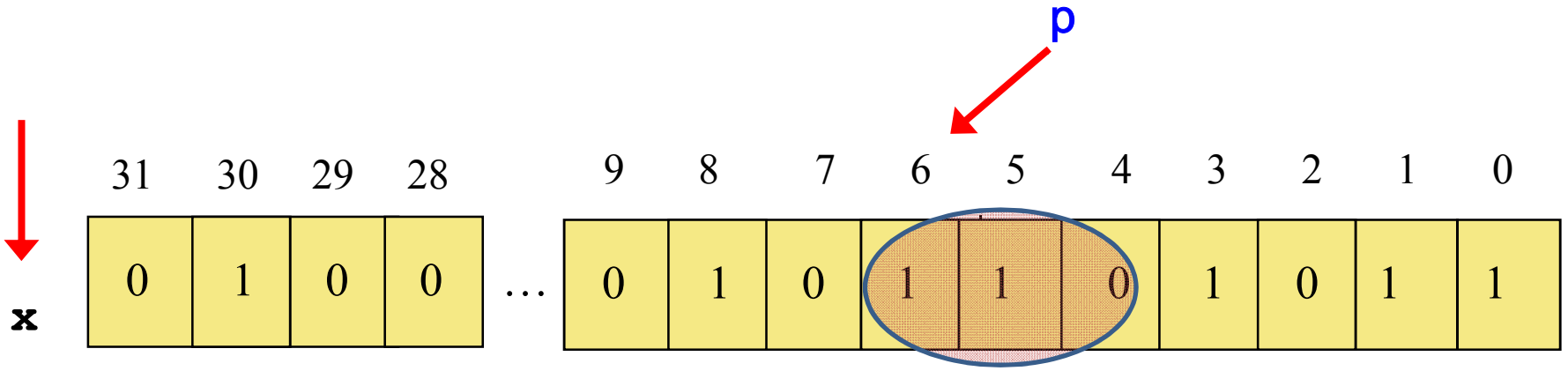
Write down a function invertBits(x,p,n) that will **invert** n bits of the integer x starting from position p leaving other bits unchanged. Assume 0 ≤ p ≤31 and n ≤ p+1



invertBits(x, 6, 3)

invertBits(x,p,n)

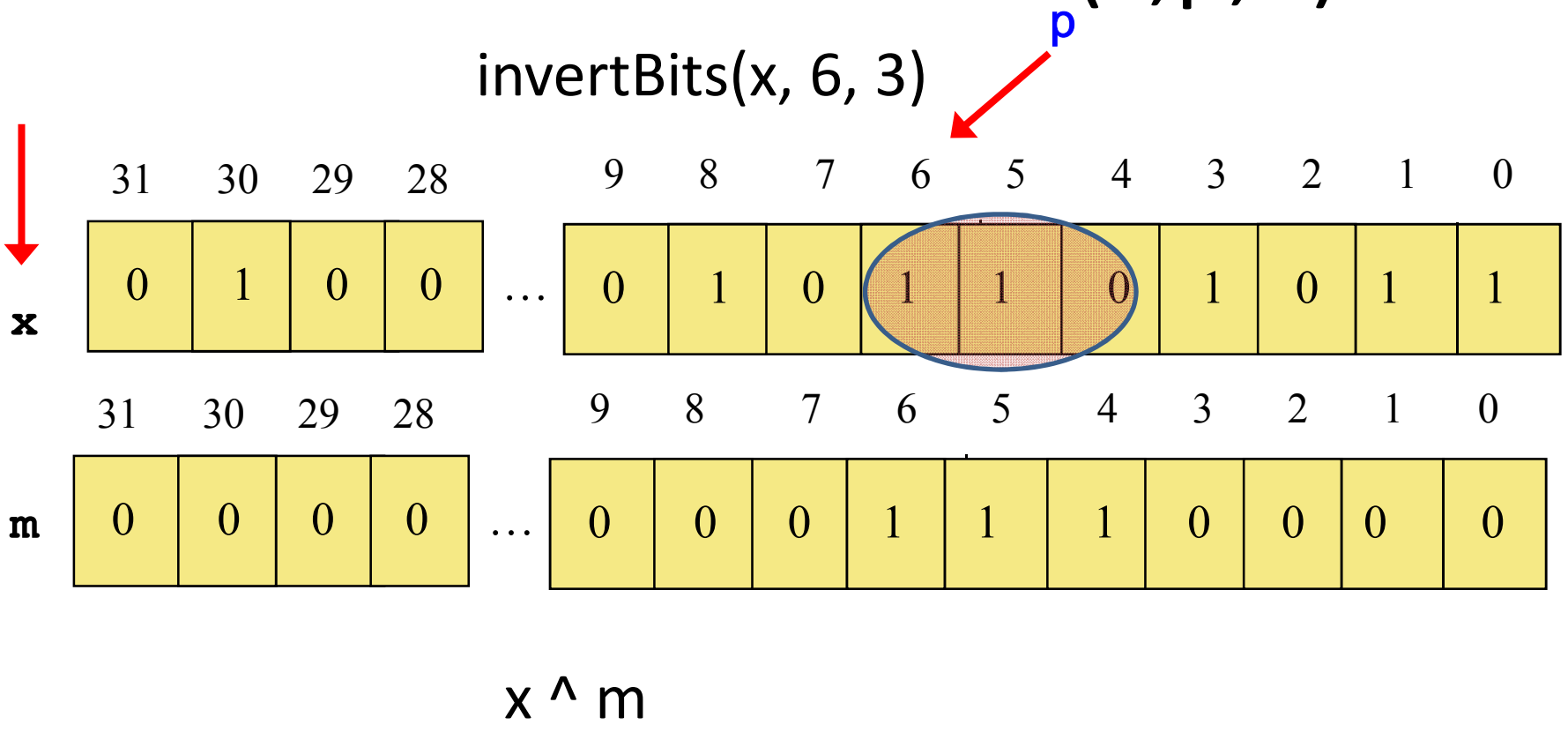
Write down a function invertBits(x,p,n) that will **invert** n bits of the integer x starting from position p leaving other bits unchanged. Assume 0 ≤ p ≤31 and n ≤ p+1



inverBits(x, 6, 3)

Call invertBit(x,p) in a loop

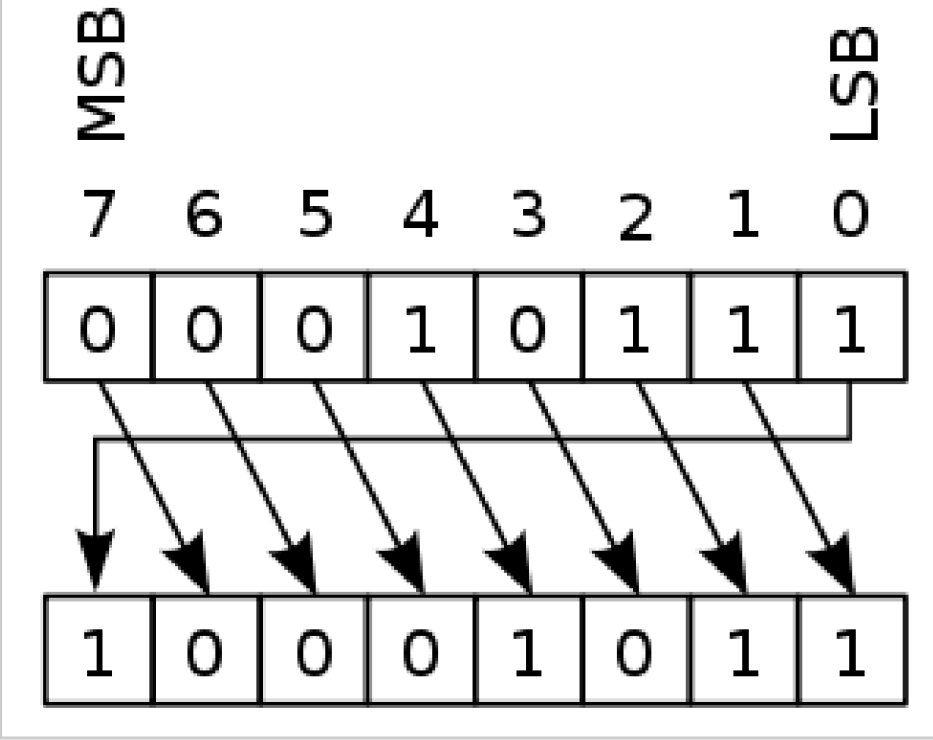
Efficient invertBits(x,p,n)



x ^ m

RIGHT CIRCULAR SHIFT

- to the right would yield: 1000 1011.



Right circular shift.

```
int circularRightShift(int x){
      unsigned int y = x;
      int s = 8*sizeof(int);
      return (y >> 1) | x << (s-1);

}
```

rightRotate (x, n)

xtractRightMostBits(x,n)

Write down a function xtractRightMostBits(x,n) that will **return rightmost** n bits of the integer x.
Assume 1 ≤ n ≤32