R-1.12

The following remarkably short and tricky function determines whether a nonnegative integer i is a power of 2.

```
bool isTwoPower(int i) {
  return (i != 0) && (((i−1) & i) == 0);
}
```

The function makes use of the fact that the binary representation of a of $i = 2^k$ is a 1 bit followed by k 0 bits. In this case, the binary representation of $i-1$ is a 0 bit followed by i 1 bits. Thus, when we take the bitwise "and" of the two bit strings, all the bits cancel out.

$$\begin{aligned} i = 1024_{10} &= 0000100000000000_2 \\ i-1 = 1023_{10} &= 0000011111111111_2 \\ i \;\&\; (i-1) &= 0000000000000000_2 \end{aligned}$$

If i is not a power of 2 and $i > 0$, then the bit strings for i and $i-1$ share at least one bit in common, the highest order bit, and so their bitwise "and" will be nonzero. We need to include a special check for zero, since it will pass this test but zero is not a power of 2.