```
/* bitOr can be implemented using only ~ and & by applying DeMorgan's Law to x OR y */
int bitOr(int x,int y) {
	return ~(~x&~y);
}

/* bitOddParity can be implemented by adding every bit to every other bit.  Since we only care
whether the final result is odd or even, we are only concerned with the value of the least
significant bit each time we add.  Using bitwise XOR to perform the additions allows us to add
multiple pairs of bits at a time
*/
int bitOddParity(int x) {
	int x16 = (x >> 16) ^ x; //bitwise XOR the top 16 bits with the bottom 16 bits of x
	int x8 = (x16>>8) ^ x16; //add the pairs of bits from the result of the previous step
	int x4 = (x8>>4) ^ x8;//add the pairs of bits from the result of the previous step
	int x2 = (x4 >> 2) ^x4;//add the pairs of bits from the result of the previous step
	int x1 = (x2 >> 1)^x2;//add the pairs of bits from the previous step
	return !(x1&1); //the even parity bit is in the least significant bit at this point, so mask
			// all the other bits and negate to get the odd partity bit.
}
```