2.64

```
int any_even_one(unsigned x){
    return !!(x & 0b10101010101010101010101010101010);
}
```

2.73

```
int saturating_add(int x, int y)
{
    int umax = 0xffffffff;
    int tmax = 0x7fffffff;
    int tmin = 0x80000000;

    int sum = x + y;
    int overflowflag = ((sum ^ (x & y)) >> 31 ) & 1 ;
    int positiveflag = !((x>>31) & 1);

    return (~(!overflowflag + umax) & sum) | (~((overflowflag & positiveflag) +
umax) & tmax) | (~((overflowflag & !positiveflag) + umax)& tmin);
}
```

2.81

A. 不总是为1，反例：x=TMin, y=0时, (x>y)为假；而-x仍为TMin，(-x<-y)为真，因此整个表达式不为1

B. 恒为1，x, y乘以常数实际上会处理为x,y的移位相加减，而加减的交换顺序不影响最终的值（即使有溢出），因此整个表达式恒为1

C. 恒不为1，原因是~x+~y = - x - 1 - y - 1 = -(x+y) - 2 = ~(x+y) - 1，反例可以取x=y=1，左式为-4，右式为-3

D. 恒为1，$-(y-x)=2^{32}-(y-x)=2^{32}-y+x=2^{32}+x-y=x-y$，而另一方面unsigned和int进行加法时均是按位相加，没有区别，故$(int)(ux-uy)=x-y$，因此原表达式恒为1

E. 恒为1，如果最低位为0，那么左移再右移和原来的二进制数码一样，因此原数一样；如果最低为为1，那么左移再右移，最低位的1变为0，无论原数正负都会比原数小1；结合两者则小于等于号成立