DataLab Lab0

1. bitXor 这个题目要求用~ &来实现异或操作，实现为~(~x&~y)&~(x&y)。首先，异或操作是对于某一位，x!=y时为1，反之为0，那么有三种情况

1. x=0,y=0

2. x=0,y=1 or x=1,y=0

3. x=1,y=1

情况1：x，y均为0时，~x&~y=1，~(x&y)=1，~(~x&~y)&~(x&y)=0

情况2：x，y有且仅有一个为0时，~x&~y=1，~(x&y)=1，~(~x&~y)&~(x&y)=1

情况3：x，y均为1时，~x&~y=0，~(x&y)=0，~(~x&~y)&~(x&y)=0

而从明面上来理解就是，使得具有相同数字的位变成1和1再进行检测

1. tmin 这个题目要我们返回二进制最小值，令1左移31位即可，获得符号位为1，其他位为0的二进制最小值
2. isTmax 这个题目要求我们检测一个输入的数字是不是二进制最大值，首先第一步，int y=x+1，如果输入的x是tmax，那么这一步的y会得到tmin，若y为-1，则获得到0，下一步，return !(y^~x)&!!y;，如果y为tmin或者0，那么y^~x得到1，!y为1，当且仅当y不为0 时，!!y不为0
3. allOddBits 这个题目要求检测奇数位上的数是否为1，第一步，我们构造一个奇数位全为1的数int test=0xAA+(0xAA<<8); test=test+(test<<16); ，接下来return !((test&x)^test);，test&x排除偶数位的影响，即将偶数位置全设为0，最后和test异或，如果全为1，那么异或后为0，取反非为1
4. negate 这个题目要求用! ~ & ^ | + << >>实现取负，常识return ~x+1;
5. isAsciiDigit 这个题目要求检测输入是否为数字0到9的阿斯克码值，思路很简单，通过和临界值作差即可isInRange = !((x + ~lower + 1) >> 31) & !((upper + ~x + 1) >> 31)
6. conditional 这个题要求我们通过! ~ & ^ | + << >>实现x ? y : z，那么首先我们要对x进行处理，即进行两次取非操作，这样可以把x转化为0或者1，那么下一步x=~x+1，如果x为0则变成全0，反之为全1，这样我们得到了一个掩码x，最后，我们只需要返回(x&y)|(~x&z)，如果输入的x为0，那么此处x为全0，x&y为全0，~x&z为z，反之亦然。
7. isLessOrEqual 这个题目要求我们用! ~ & ^ | + << >>实现判断x <= y是否成立，本题的关键在我看来是在于处理int类型的溢出问题，我们如果仅仅简单地对两个数作差的话，可能会产生溢出，导致判断错误，这就需要我们分情况来讨论了。首先ifSignDiff=((x^y)>>31)&1，这一步用来检测xy的符号是否不同，下一步SignOfX=(x>>31)&1，SignOfY=(y>>31)&1，这两个操作分别获得了xy的符号位，下一步SignOfDiff=(y+(~x+1))>>31&1，这是获得y-x的符号，注意这里可能发生溢出，那么最后返回(!ifSignDiff&!SignOfDiff)|(ifSignDiff&SignOfX)，共有两种情况符合要求，第一种为符号相同，此时不会发生溢出，若y-x为非负数即可满足x<=y，第二种为xy符号不同，此时有且仅有x为负数是符合，直接检测x符号即可。
8. logicalNeg 这个题要求我们用~ & ^ | + << >>实现逻辑非的操作，第一步，y=~x+1，这样我们就获得了-x，下一步sign=(x|y)>>31，首先x与y按位或，如果x不为0，则结果的符号位一定是1，那么右移31为后获得全1，最后返回sign+1即可，即0，但是如果输入的x为0，那么返回的是0+1=1
9. howManyBits 这个题要么计算二进制表示一个输入的数字需要多少位，我们发现，16 8 4 2 1 的任意组合恰好可以组成1~32的任意数字，利用这个二分的方法来构造算法，首先计算是否有16位，在算剩下的是否有8位，以此类推。
10. floatScale2 这个题目要我们计算输入值\*2，那么首先exp=(uf&0x7f800000)>>23，这一步获得符号位，其中0x7f800000为浮点数最大值，指数部分全1，其他部分全0，这样以获得指数部分，下一步sign=uf&(1<<31)，这一步的作用为获得符号位置，即利用符号为1，其他部分为0的数字，与uf按位与，这一就获得了符号为为0或者1，其他的部分为全0的数字，那么这个时候我们检测exp，如果exp为0，则说明这是一个非规格化的浮点数，直接左移一位即可，然后加上符号位，这是为了避免如果原本符号位为1，左移导致1变为0导致变号。那么如果不是的话检测是否为255，如果exp==255，这说明输入的uf为无穷大或者为nan，直接返回即可。上述情况都是不是的话我们将exp左移一位，即\*2操作，这时候检测exp是否为255，如果加一后exp变为255，说明到达最大值，给定符号输出最大值即可。那么最后一张情况，将exp恢复相应的位置，利用0x807fffff，即符号位为1，指数为全0，尾数为全1，和uf按位与可以获得uf的符号以及尾数，最终利用按位或将exp部分组合上去。
11. floatFloat2Int 这个函数要求我们实现将单精度浮点数表示的位级别等效整数值的函数。首先提取符号位、指数位和尾数位，那么接下来对指数进行检测，如果指数为0xFF（全1），或者大于0x96，则表示特殊情况，包括NaN和无穷大，函数返回0x80000000u表示无效结果。接下来计算偏移量，即将指数减去127，此时如果指数小于0，则说明是小数，返回0，如果大于等于31，则溢出，返回0x80000000u表示无效结果，那么最后，将尾数的最高位置为1，然后将其左移或右移以匹配指数的大小，返回是，考虑符号位，若之前记录的符号位为1，则返回-result，即返回一个负数。