Lab1实验报告

Final score



Update:2024.12.1 16:32



当然也可以

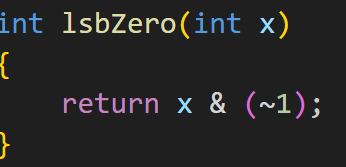


郑重声明：

解题方法均为本人自己想的（可能不是最优），除了浮点数学习了一下（但也是自己优化的，从15优化到13）。

倒着食用更香

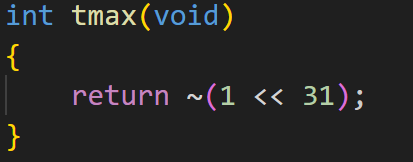
T1



水，无需多言

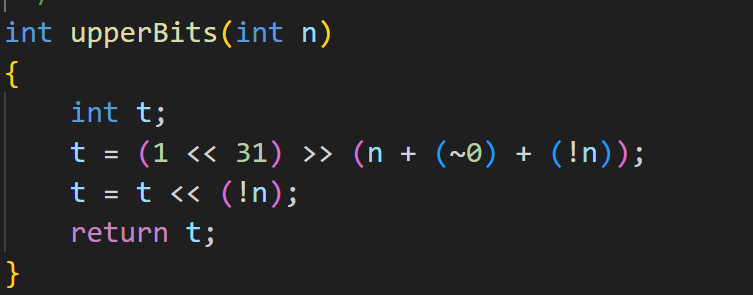
(x>>1)<<1同理

T2



水，无需多言

T3



这题有点含金量

先移到最高位，然后在运用算术右移就行了

但是要右移n-1位

-1可以表示为(~0)

但因为n可以为0

不能右移-1位

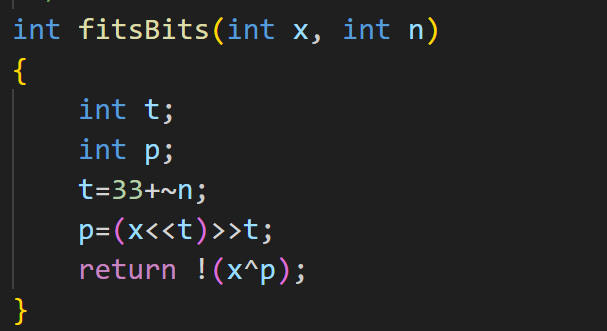
所以当n为0时

先加个1

然后再左移1就好了~

然后1用!n代替就好了

T4



一般

简而言之就是把n位x做个符号扩展，然后看和原数是不是一样就好了

当前x的最高位为n-1

要把它移到31位，

要移动32-n位

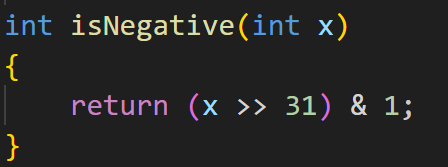
-n就是 (~n)+1

就完事了

然后再移回来就符号扩展了

最后和原数异或后，看是不是0，是0就返回1，不是就返回0，用!就完事了

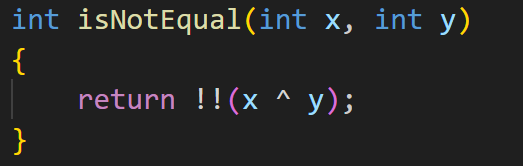
T5



水

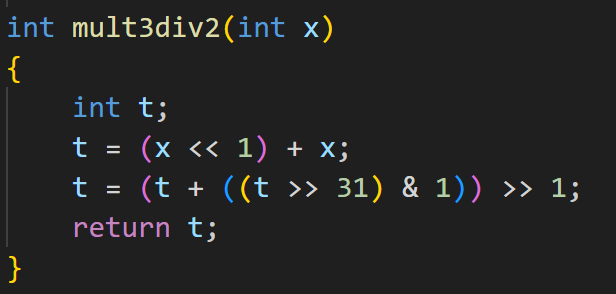
取个符号位就完事

T6



水

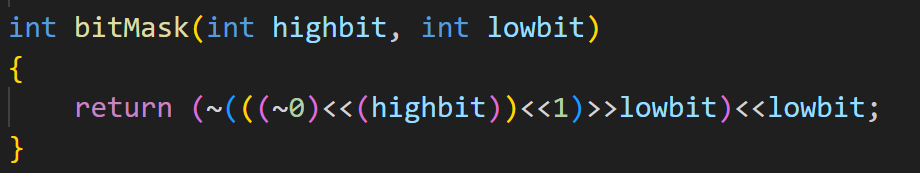
异或后判是不是零就完事了

T7 

一般

注意负数除的时候要加个偏置量就可以了

T8



有点东西

因为highbit==31时，1<<(highbit+1)会等于1

但1<<32等于0，估计变量在移的时候会取模32

然后这题思路真难讲

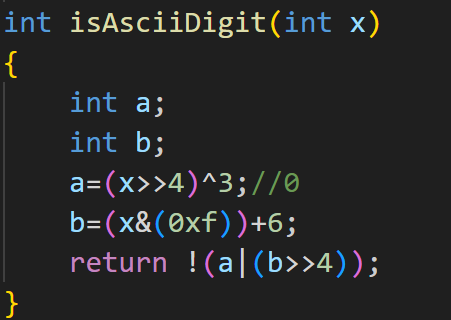
最开始是0xffffffff

然后左移highbit+1位之后会把0~highbit位全变0

再取反后，0~highbit位全变1，其它是0了

然后再把0~lowbit位变0就好

T9



本题有点东西

随便写写都能过

但是要优化到比老师少，还是想了好久

本题就是判0x30 <= x <= 0x39

方法有很多

上图做法是这样的

a就是先判是否是0x30??

b的话，用了一点特殊方法

b是先取低四位，然后加6，看会不会溢出

为什么会这样想呢

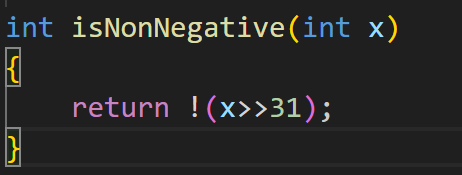
b取完低四位后就取值0-15

但是我们只要0-9

所以只要判加6会不会溢出到第五位就好了

最后return偷懒了，我们要的是a==0和(b>>4)的情况，正常多几个判断符也可以，但是！(a|(b>>4))是最少的，因为当其中有一个不为0都会返回1（花点时间理解？）

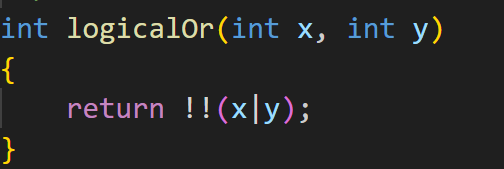
T10

. 

水

看有没有符号就行了

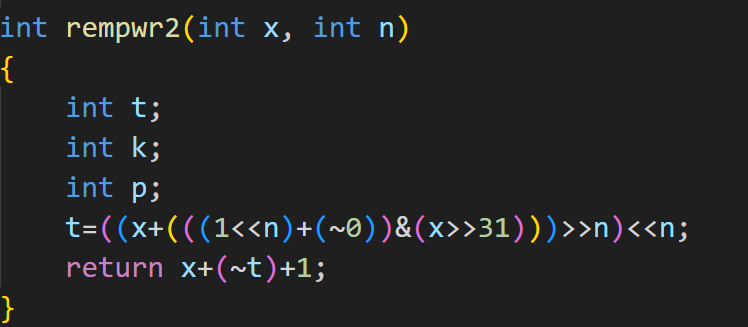
T11



水，无需多言

只要看两个数或完是不是0就完事了

T12



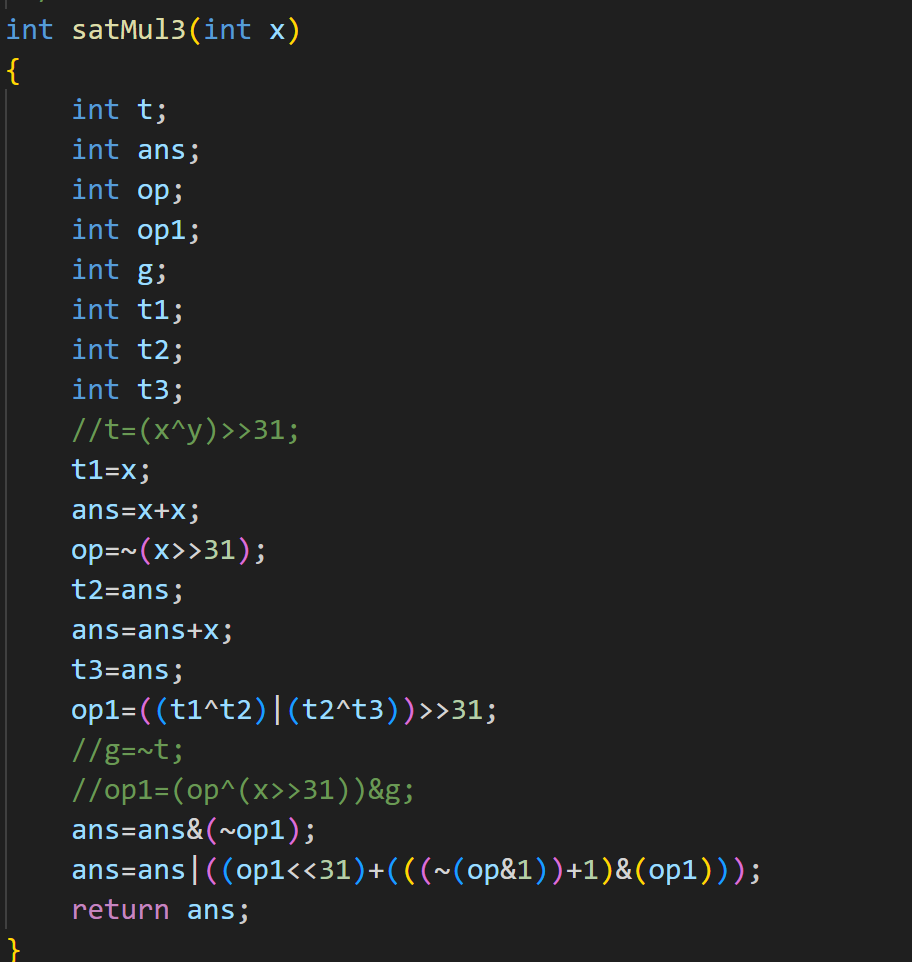
一般

用到一点取模的基本知识



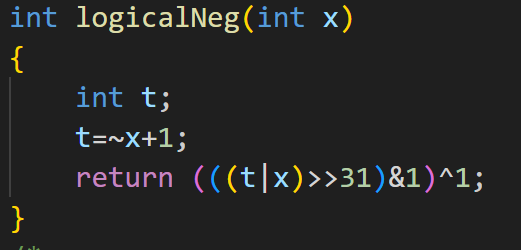
然后直接用就好了吧（这还看不懂真没办法了。。。）

T13



和下文satAdd一起讲，性质一样

T14



本题有点卡我思路

后面奇奇怪怪想到的

就是x==-x时

才是零

但是0x8000000也会相等

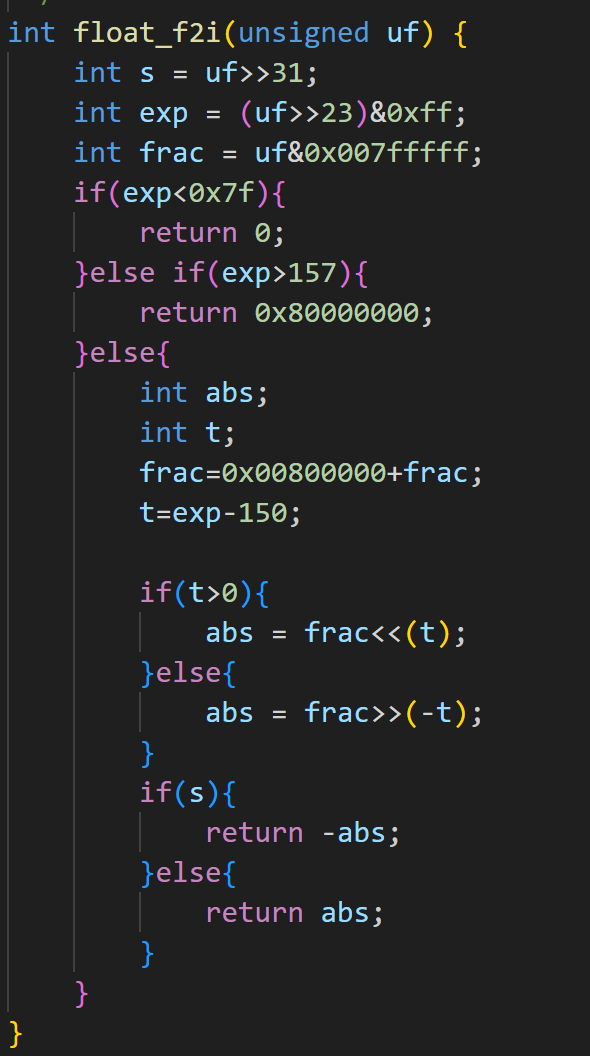
但是0的话两次符号位都是0

其它的除0x8000000是两次，都是一次

所以我们只要把x和-x符号位拿来看看就好了

这边没有用 (((t>>31)|(x>>31))&1)^1是因为可以省一个（省的思路可以自己脑补一下，再写就太多了）

T15



就是把浮点数各位拿出来然后运算判断就完事了

s是符号位

exp是阶码

frac是位数

第一个判断exp<127时

你后面的尾数都没了，所以直接返回0就好

当然这个判断同时也把非规格数给判掉了，就不用考虑直接返回0就好

第二个判断是判断特殊值同时还有溢出的规格数

1.1.????规格数左移31位会溢出，所以exp-127>30就会爆，也就是exp>157

2.当exp是8位1时就返回题目给的特殊值

第三个判断就是计算规格数

因为浮点数的表示是1.???

然后你就要把1搞回去

同时exp还要减掉127

那为什么减掉的是150呢

因为我们把阶码算成整数了

相当于小数点右移了23位

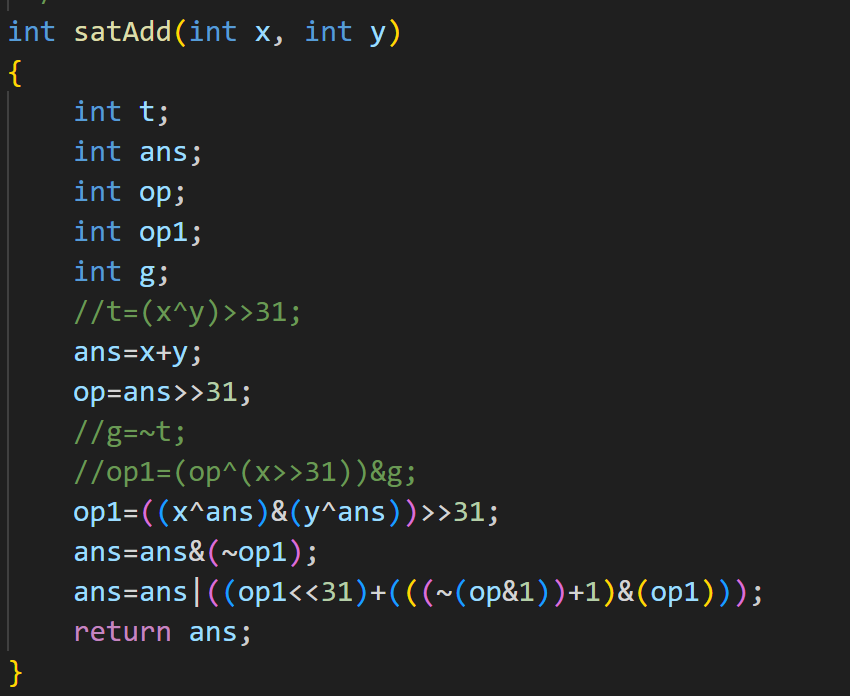
所以阶码还要-23就行

然后就判正负移就完事

最后记得把符号搞回去就行

（当时以为要舍入，结果不用。。。）

T16



和上面的satMul3一起讲

有几个变量要说明一下：

op代表的是与原数相反的符号

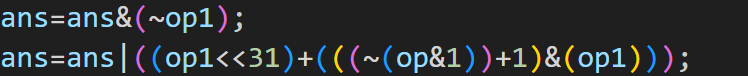
op1是代表溢出的标志

以上两者要么是全0要么是全1

ans是最终的答案

当然ans最初赋个原始操作的结果

代码中最重量级的部分在下面



（我真的是好佩服自己，能想到这么天才的代码）

第一行是清零操作

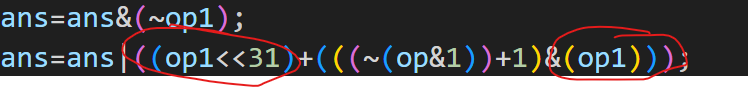
当op1是全1时代表有溢出

然后我们把原数清空准备放新数

反之op1为0时~op1就是全1，然后与完ans不变

第二行是放数操作

当op1是1的时候才放



当op1是0后，你会发现上面的操作不管op怎么样就是0

ans或完0后没影响

当op1是1后，然后分op讨论：

当op是全0时代表我们放的数应该是0x8000000

所以直接(op1<<31)就能拿到

当op是1时代表我们放的数应该是0x7fffffff

然后合理运用下溢，直接减1就好

怎么减1呢

我们把op的最低位取出后然后各位取反末尾加1就行

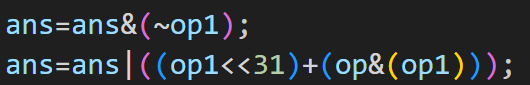
为什么是op的最低位呢

当op全1时取出的数是1，然后才能变成-1

当op全0时取出的数是0，然后运算后是0不影响

最后&(op1)是为了防止没有溢出ans还与上一个数就爆了

Update：2024.12.1 15:55



有位大佬发现 (~(op&1)+1)可以直接优化为op确实有道理

一下可以省3\*2个运算符

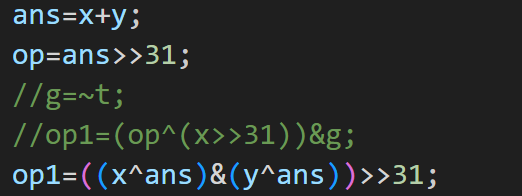
当op是全一时，我们要的是-1，那不就是op吗？

当op全0时，我们要的是0，那也是op呀

所以直接用op就好

核心代码已实现完成

怎么取op1和op呢



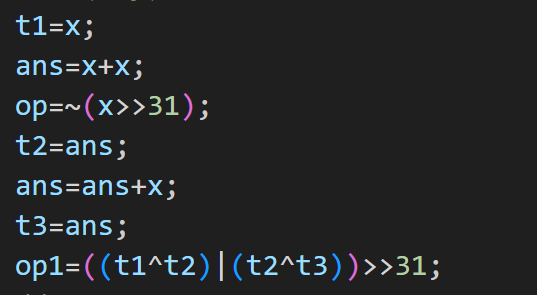
op其实很好取，~(x>>31)也可以

这边为了省运算符所以直接这样搞

op1呢

直接看ans分别与x和y的符号是不是相反就行

satMul3中

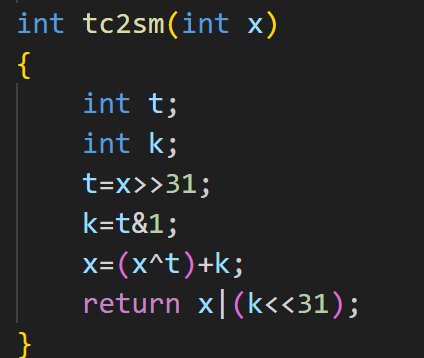


op就直接~(x>>31)就行

op1就直接每次累加x后看看符号有没有变就行

（在这里省符号搞了好久。。。。）

T17



一般

先取符号位就好了

然后是负数就各位取反后加1就行了

用到一点小技巧，x^t

t是符号位，要么全1要么全0

这边x^t只有t是全1才会取反

最后再把符号位搞上去就完事了