

Datalab

第一题

bitXor

only use ~ and | to make x&y

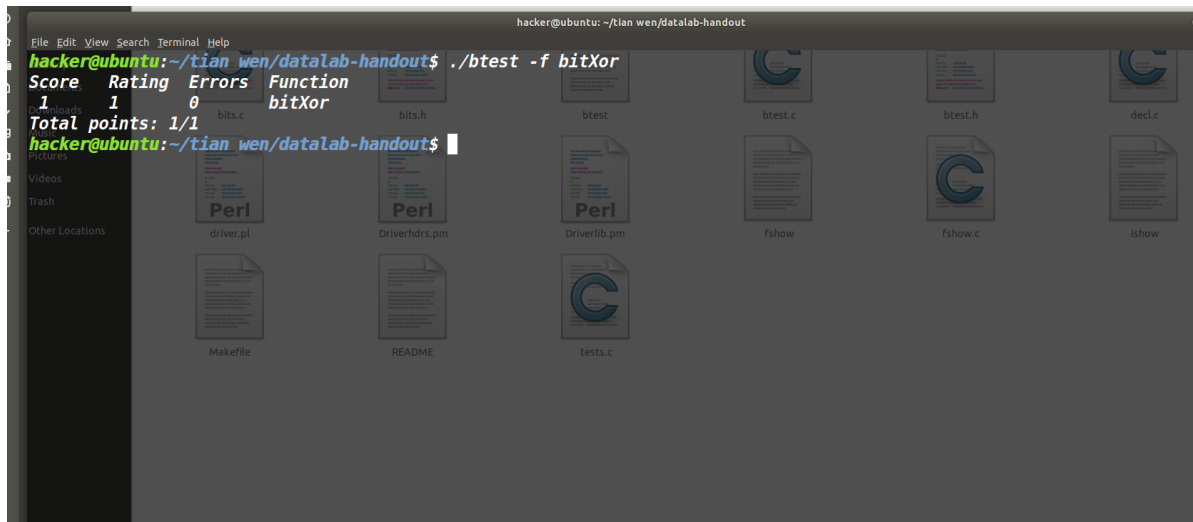
用~和| 来实现与运算、

比较简单离散学过了

```
int bitXor(int x, int y) {  
    return ~(~(x&(~y))&(~((~x)&y)));  
}
```

给了一个example样例

测一下

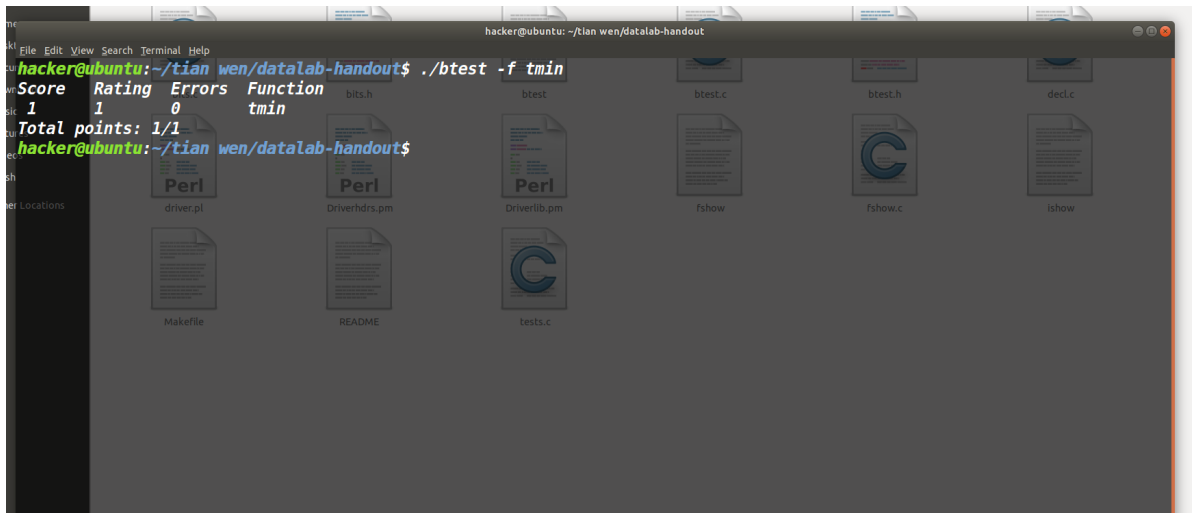


第二题

tmin

返回最小的int值，也就是让他溢出 $1 \ll 31$ 这样就是最小的int值了。

```
int tmin(void) {  
    return 1<<31;  
}
```



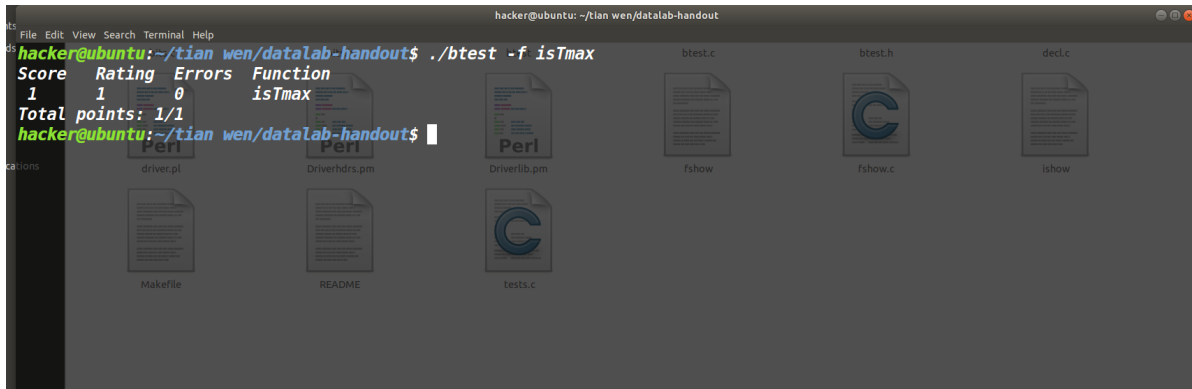
第三题

判断x是否是最大的int值

主要想法是溢出后*2为0，但同时满足的还有0

所以考虑排除0即可

```
int isTmax(int x) {
    int t=x+1+x+1;
    //printf("%d",!t);
    //printf("%d\n",q);
    int result=(!t)&(~(! (x+1)));
    return result;
}
```



第四题

```
int allOddBits(int x) {
    int format=0xAAAA;
    int qq=format+(format<<16);
    return !((x&qq)^qq);
}
```

为了判断是否只有奇数位上面是1，

那么就先搞个format 然后&一下 就只会比 format 的1个数还少而且1也都是奇数位的


然后再^下就知道是否和原来的format一样，一样的话就代表只有奇数位有1 其他无。否则就是其他位也有。

第五题

```
int negate(int x) {
    int q=~x+1;
    return q;
}
```

取负数

不多说了



```
(root@DESKTOP-PQTH8BD) ~/mnt/.../Desktop/天问/第一周csapp/datalab-handout
# ./btest -f negate
Score  Rating  Errors  Function
2      2      0      negate
Total points: 2/2
```

第六题

判断是否x在区间内

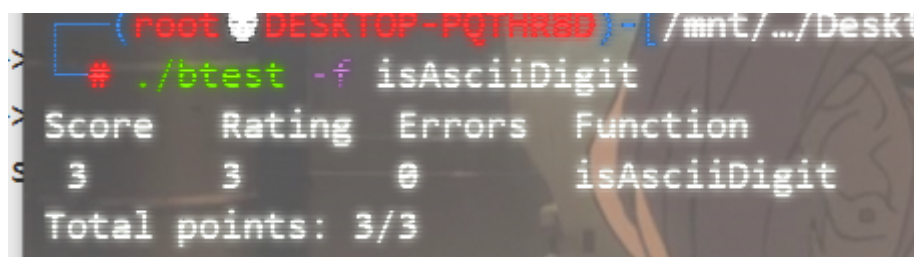
先找出Bit之间的规律。

然后找到了00111001 00110000

之间所有数

```
int isAsciiDigit(int x) {
    int a=3;
    int ss=(x>>6);
    int tt=(x>>4)&1;
    int ee=(x>>5)&1;
    int q=tt&ee;
    //printf("%d\n",q);
    int aa=(x>>3)&1;
    int bb=(x>>2)&1;
    int cc=(x>>1)&1;
    return (!ss)&q&((!aa)|aa&(!bb&(!cc)));
}
```

一个大判断.



```
(root@DESKTOP-PQTH8BD) ~/mnt/.../Desk
> # ./btest -f isAsciiDigit
> Score  Rating  Errors  Function
s 3      3      0      isAsciiDigit
Total points: 3/3
```

第七题

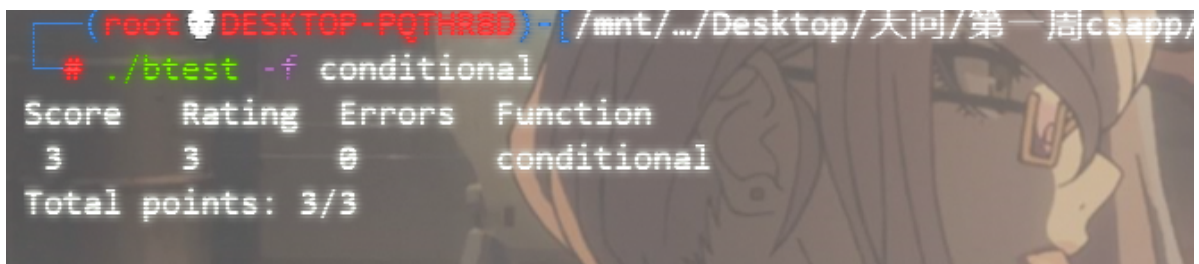
实现三元运算符

经过查 发现任何数字 $\&(-1)$ 都是本身

那么通过这点可以构造

让一边为0另一边为自己然后或就可以了

```
int conditional(int x, int y, int z) {
    int mask = ~0 + !x;
    //printf("%d\n", !x);
    //printf("%d", mask);
    return ((mask&y)|(~mask&z));
}
```



```
(root@DESKTOP-PQTH889) ~ [~/mnt/.../Desktop/天问/第一周csapp/
# ./btest -f conditional
Score  Rating  Errors  Function
3      3      0      conditional
Total points: 3/3
```

第八题

可以想到 $x-y < 0$ 这一点

来判断 大小

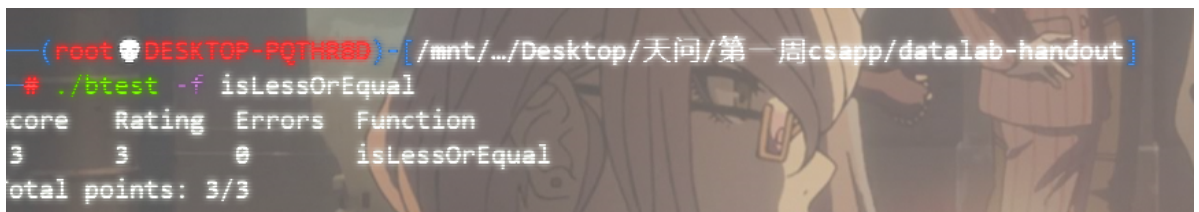
但是还要考虑溢出问题。。

调了很久。。。

通过 $\gg 31$ 来判断符号

```
int isLessOrEqual(int x, int y) {
    int q=(~x+y+1)>>31;
    int a=(x>>31)&!(y>>31);
    //printf("%d\n",a);
    int b=!(x>>31)&(y>>31);
    return a|((!q)&(!b));
}
```

只有 x 不是负数 y 是负数 或者 $y > x$ 并且 x 不是正数 y 不是负数情况下返回1。（好像写反了） $x \gg 31$ 为1 是负数



```
(root@DESKTOP-PQTH889) ~ [~/mnt/.../Desktop/天问/第一周csapp/data-lab-handout]
# ./btest -f isLessOrEqual
Score  Rating  Errors  Function
3      3      0      isLessOrEqual
Total points: 3/3
```

第九题

实现!

非0则1 是0则0

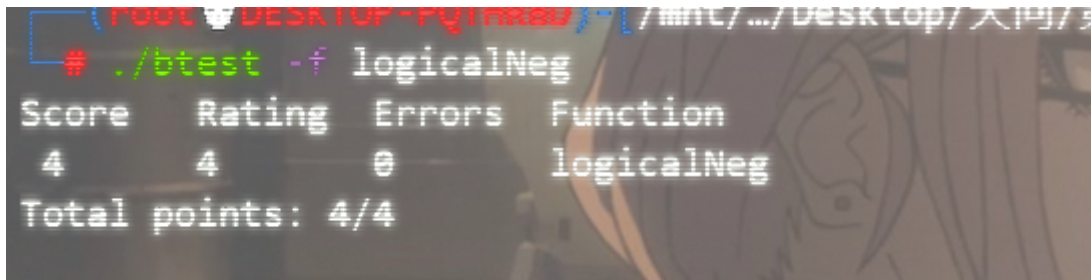
```
int logicalNeg(int x) {
    return ~(~((x|~x+1)>>31))+1;
}
```

搞了一个贼奇怪的嵌套。。。

先是找了个办法让其有办法使得非0数和0区别出来即

$\sim x + 1 | x$ 后 $\gg 31$ 必为 1

而0却是 0



第十题

查有多少位

u1s1我想枚举。

反正操作数肯定够

这里写下学的二分吧。。

```
int howManyBits(int x) {
    int sum=0;
    x=x^(x>>31);
    //printf("%d",x);
    sum+=(!!(x>>16)<<4);
    x=x>>((!!(x>>16)<<4));
    //printf("%d",x);
    sum+=(!!(x>>8)<<3);
    x=x>>((!!(x>>8)<<3));
    // printf("%d",x);
    sum+=(!!(x>>4)<<2);
    x=x>>((!!(x>>4)<<2));
    // printf("%d",x);
    sum+=(!!(x>>2)<<1);
    x=x>>((!!(x>>2)<<1));
    // printf("%d",x);
    sum+=(!!(x>>1));
    x=x>>((!!(x>>1));
    sum+=x;
    return sum+1;
}
```

```
(root@DESKTOP-PQTH88D) - [ /mnt/.../Desktop/天问/第
# ./btest -f howManyBits
Score Rating Errors Function
4 4 0 howManyBits
Total points: 4/4
```

第十一题

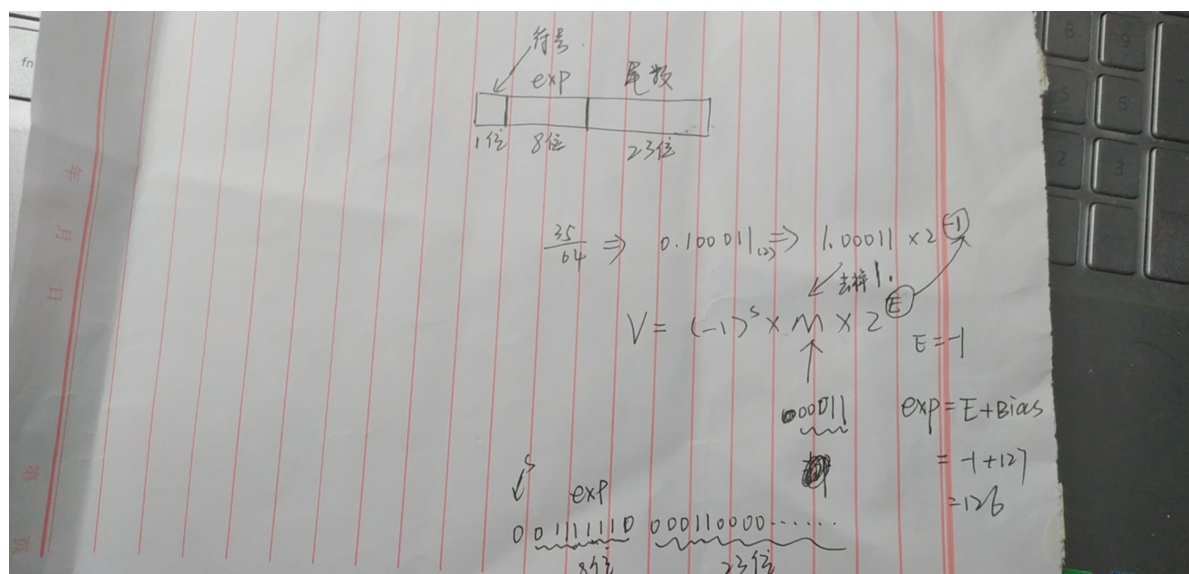
浮点数运算

也就是 $\times 2$

```
unsigned floatScale2(unsigned uf)
{
    int exp=(uf&0x7f800000)>>23;
    int sign=uf&(1<<31);
    if(!exp)
        return uf<<1|sign;
    if(exp==255)
        return uf;
    exp++;
    if(exp==255)
        return 0x7f800000|sign;
    return (exp<<23)|(uf&0x807fffff);
}
```

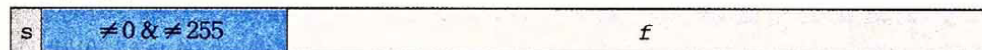
分析下float如何存储

以及254 的时候是可以

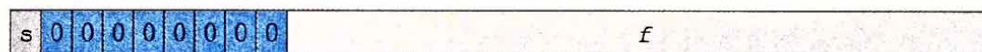


所以当exp为0的时候直接 $\times 2$ 即可。若为255 则是NAN或者 ∞ 返回原值。

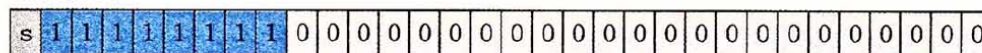
1. 规格化的



2. 非规格化的



3a. 无穷大



3b. NaN

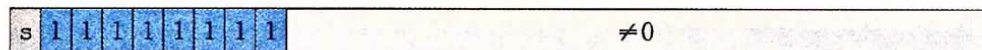


图 2-33 单精度浮点值的分类(阶码的值决定了这个数是规格化的、非规格化的或特殊值)

然后最后还原回去。就可以了。

```
(root@DESKTOP-PQTHR8D) - [ /mnt/.../Desktop/天问/第一周csapp/data1
# ./btest -f floatScale2
Score Rating Errors Function
  4      4      0 floatScale2
Total points: 4/4
```

第十二题（咕咕下）之后研究

第十三题

```
unsigned floatPower2(int x) {
    int exp=x+127;
    if(exp>=255) return (0xFF<<23);
    else if (exp<=0) return 0;
    else return exp<<23;
}
```

2.0^x 。也就是直接+127 如果溢出还是覆盖1111 1111

如果小于等于0 就直接返回0

否则返回正常得exp。