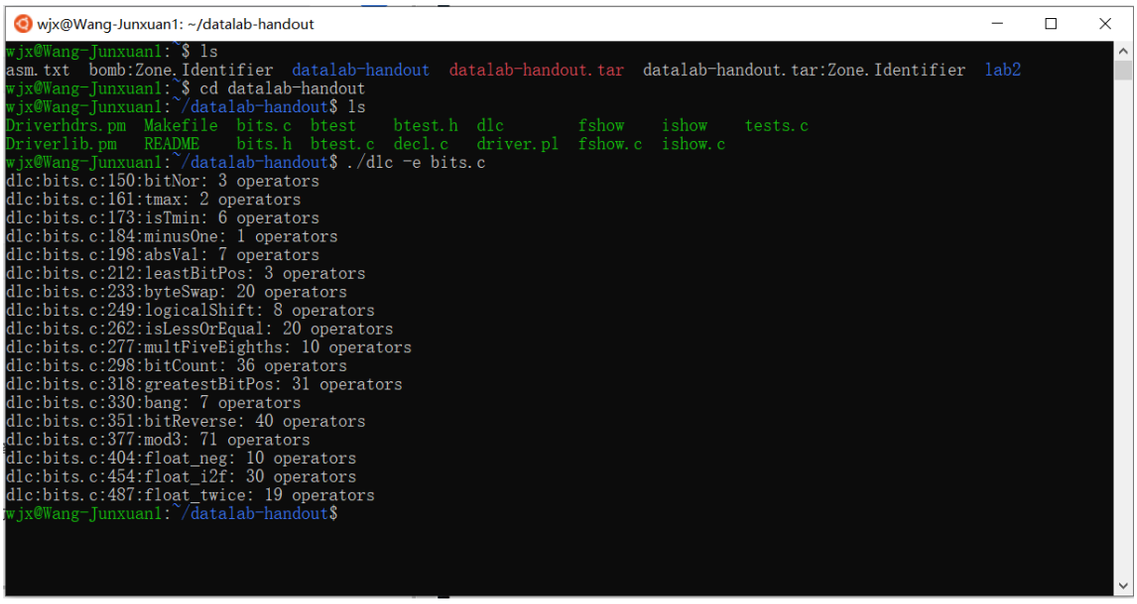
Datalab

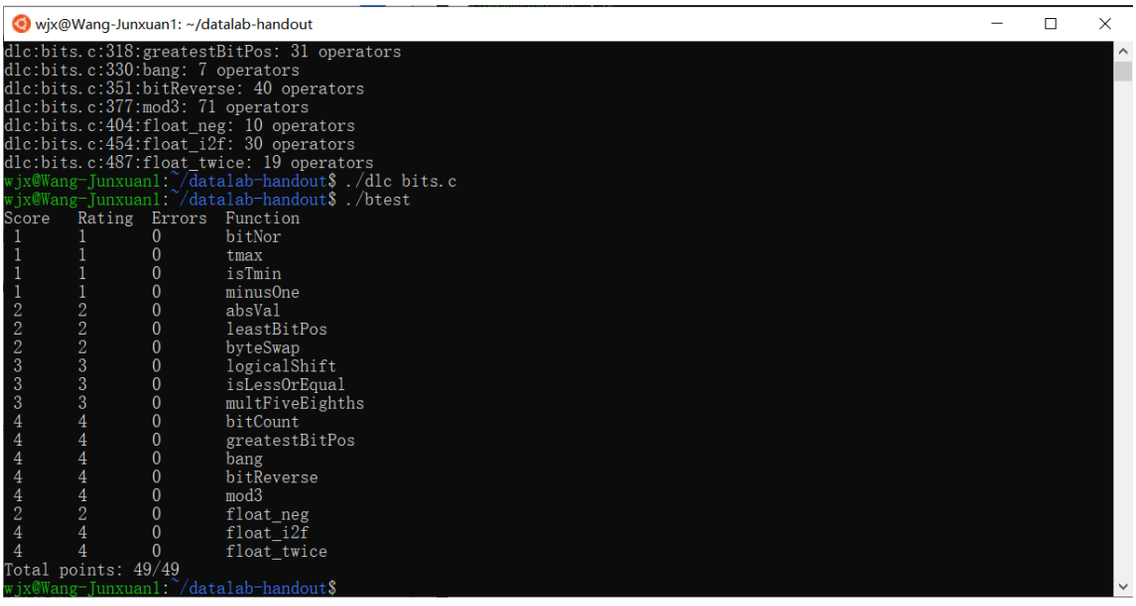
姓名：汪俊轩

学号：21307130169

终端执行 ./dlc -e bits.c后的截图



终端执行./btest后的截图



每个函数的思路：

bitNor（）

通过分析每一位的真值情况即可获得，只在全为0时结果为1。

tmax（）

最大的数除了符号位其他全为1，只需要得到符号位为1的数再取反即可。

isTmin（）

只有最小的二进制数和0满足x^(~x+1)为0，利用和！x位或排除0的情况之后再取反即可。

minusOne（）

-1是所有位都为1的数，故对0取反即可。

absVal（）

取出符号位用于判断是否取反加一，表达式设为((~flag)&x)|(flag&((~x) + 1))，用到了flag是否为0来调节究竟是返回x还是((~x) + 1)。

leastBitPos（）

想要得到最后一位1，应该进行某些只会影响到最后一位1的操作，经过思考后发现取反+1的操作会影响到最后一位1的位置，完善后可得((~x) + 1)&(x)。

byteSwap（）

调换第m，n位只需要分别用只在这两位上是1的数把这两位的值提取出来，再移位后填入即可完成交换动作。

logicalShift（）

得到一个前n位全为0的数与移位后的数位与，如此把即前n为置位0。

isLessOrEqual（）

判断y-x是否非负，同时排除掉因为溢出（如y为负而x为正的情况）即可。

multFiveEighths()

可拆解为除以二加上除以八。为了防止移位的时候位的丢失，将后三位和前面29位分开处理。

bitCount（）

原本思路是一个一个相加，但如果一个一个相加，写出代码后发现符号数完全不够用，因此必须使得一个+号完成多次加法，因此想到把奇数位和偶数位相加，再把01位和23位相加（以此类推），可以做到一个+号完成了多次加法，最终得出了答案。

bang（）

本题是要将0与非0数的差别在某一位具体的数字上体现出来，因此思考后发现可以利用任何非0数都满足本身或者本身加上最大二进制数之后首位总有一个会为1的特点，将0与非0数区分开来，得到最终结果。

bitReverse()

先相邻两位调换，再相邻四位调换，再相邻八位调换，最后相邻16位调换，即可实现32位数的从头到尾的调换。

mod3（）

利用数学知识发现，从低位到高位，出现在第奇数位上的1总会给该数贡献余数1，出现在第偶数位上的1总会给该数贡献余数2，如此将它们分别相加即可化简，重复多次即可得到最终结果，最后将3转化为0即可。

float_neg()

将尾数部分取出，用if结构讨论是否需要将符号位取反即可。

float_i2f()

同样分情况讨论，现将0和最小负数的情况取出，之后再判断最高位的位置自后进行移位即可，注意一下进位。

float_twice（）

将0和最小负数情况取出后讨论即可