

运行结果：

```
问题 8 输出 测试结果 终端 调试控制台 端口 1
[build] functional-3/042_while_test1.sysu.c ..... 100.00/100.00
[build] functional-3/043_while_test2.sysu.c ..... 100.00/100.00
[build] functional-3/044_while_test3.sysu.c ..... 100.00/100.00
[build] functional-3/045_break.sysu.c ..... 100.00/100.00
[build] functional-3/046_continue.sysu.c ..... 100.00/100.00
[build] functional-3/047_op_priority5.sysu.c ..... 100.00/100.00
[build] functional-3/048_op_priority4.sysu.c ..... 100.00/100.00
[build] functional-3/049_unary_op.sysu.c ..... 100.00/100.00
[build] functional-3/050_unary_op2.sysu.c ..... 100.00/100.00
[build] functional-3/051_logi_assign.sysu.c ..... 100.00/100.00
[build] functional-3/052_stmt_expr.sysu.c ..... 100.00/100.00
[build] functional-3/053_arr_expr_len.sysu.c ..... 100.00/100.00
[build] functional-3/054_assign_complex_expr.sysu.c ..... 100.00/100.00
[build] functional-3/055_if_complex_expr.sysu.c ..... 100.00/100.00
[build] functional-3/056_short_circuit.sysu.c ..... 100.00/100.00
[build] functional-3/057_short_circuit2.sysu.c ..... 100.00/100.00
[build] functional-3/058_scope.sysu.c ..... 100.00/100.00
[build] functional-3/059_sort_test1.sysu.c ..... 100.00/100.00
[build] functional-3/060_sort_test7.sysu.c ..... 100.00/100.00
[build] functional-3/061_empty_stmt.sysu.c ..... 100.00/100.00
[build] functional-3/062_side_effect.sysu.c ..... 100.00/100.00
[build] functional-3/063_nested_calls2.sysu.c ..... 100.00/100.00
[build] functional-3/064_while_if.sysu.c ..... 100.00/100.00
[build] mini-performance/00_bitset1.sysu.c ..... 100.00/100.00
[build] mini-performance/01_mm1.sysu.c ..... 100.00/100.00
[build] mini-performance/crypto-1.sysu.c ..... 100.00/100.00
[build] mini-performance/dead-code-elimination-1.sysu.c ..... 100.00/100.00
[build] mini-performance/fft0.sysu.c ..... 100.00/100.00
[build] mini-performance/hoist-1.sysu.c ..... 100.00/100.00
[build] mini-performance/if-combine1.sysu.c ..... 100.00/100.00
[build] mini-performance/instruction-combining-1.sysu.c ..... 100.00/100.00
[build] mini-performance/integer-divide-optimization-1.sysu.c ..... 100.00/100.00
[build]
[build] task1
[build] 总分（加权）：100.00/100.00
[build] =====
[build]
[build] 成绩单已保存： /workspaces/SYsU-lang2/build/test/task1/score.txt
[build] JSON 格式： /workspaces/SYsU-lang2/build/test/task1/score.json
[driver] 生成完毕：00:00:01.277
51 13 13 生成已完成，退出代码为 0
```

实验感想：

该实验使用 **antlr** 完成。

本次实验在助教的修改下已经相当容易，在环境以及语法等方面都没有遇到较大障碍。代码整体架构很简明易懂。

在.g4 文件中实现了 RE 的编写。比较出人意料的是对于 0xb 属于 constant 的 RE 编写。按照原逻辑 0xb 被分成了 <numeric_constant,0>、<identifier,xb> 两个。最后以添加 HexConstant 定义解决。该文件中对 LineAfterPreprocessing、空格与换行的定义极大简化了编程构思与实现的难度。在 main 中可以快速利用他们实现输出。

根据提供的 StackOverflow 页面可以得知对预编译行的合理处理方式。每一行前面的数字是紧跟该行的代码在源文件中的起始行。中间是源文件地址。最后的数字则是 flag。利用 cpp 库 <regex> 编写正则表达式获取 Line 和 location。对于 StartOfLine 和 LeadingSpace，可以用设置两个对应的全局变量作为 flag 标志输出。最后用“\t”调整格式即可。