

Lab4-report-201502008

201502008 程潜

实验环境：MacOs

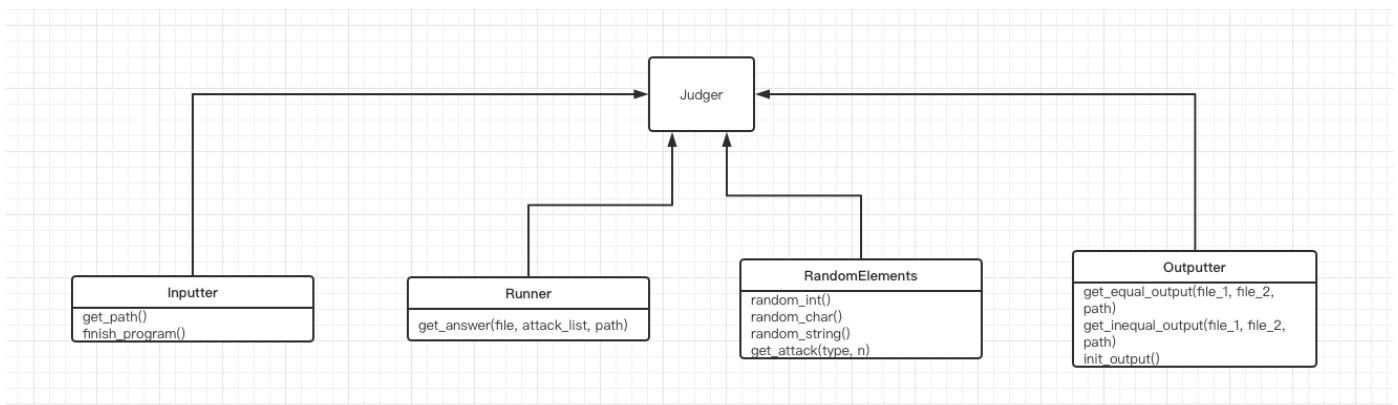
实验语言：python

IDE：VS Code

本次实验实现了简易版等价判断和确认工具并且在github上熟练运用了git相关指令完成远程开发。以下将一一介绍：

等价判断工具源码分成了五个部分，分别是处理输入inputter，处理输出outputter，主体代码judger，运行程序runner，和随机生成攻击输入的random_elements：

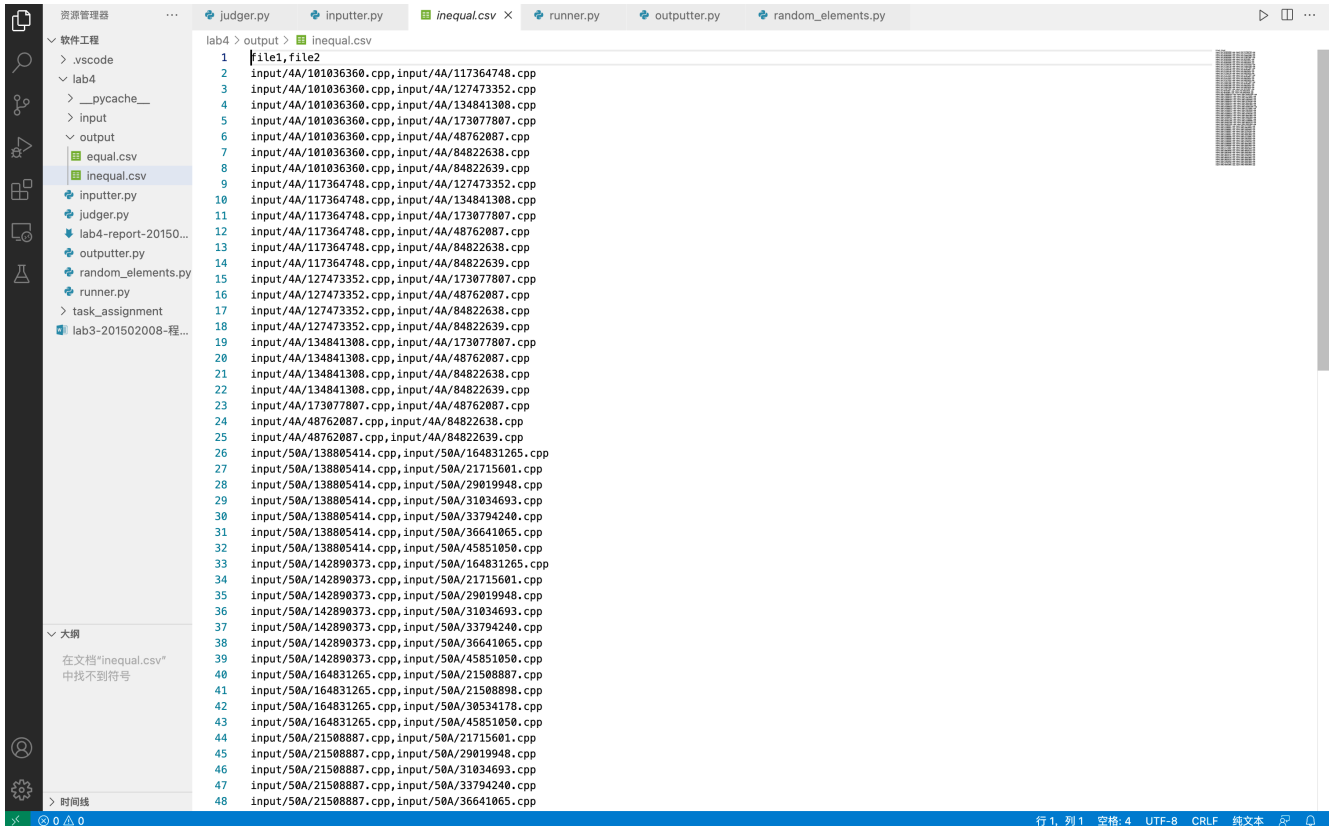
主体代码judger



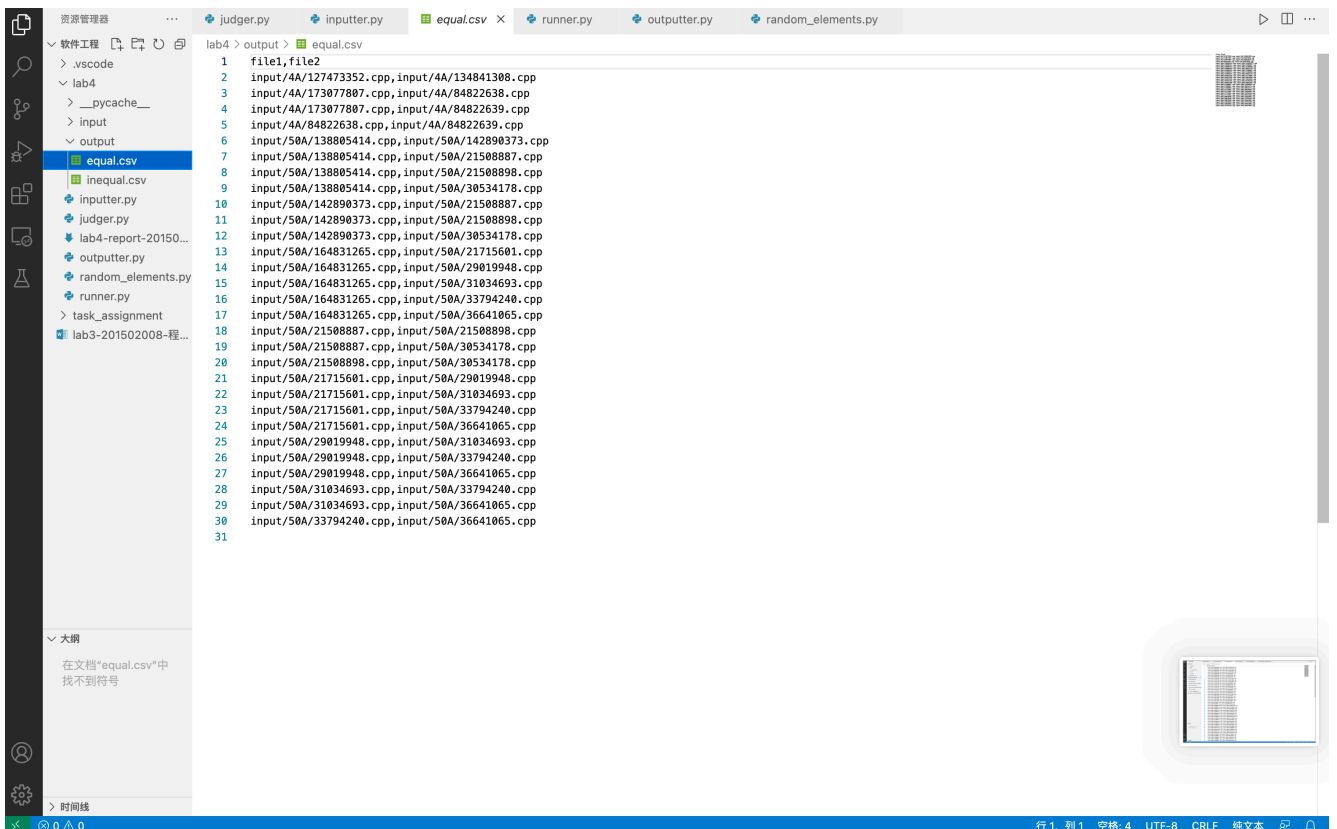
主体代码judger的主要内容就是调用各个部分并完成判断等价性，其内容主要分为以下几个步骤：

1. 通过调用inputter.py中的`get_path()`得到输入文件，然后对Input文件进行检查，只留下其中的文件夹（因为在MacOs下的文件夹中会有隐藏文件，这个操作相当于过滤掉了这些隐藏文件），从而得到多个输入文件夹。
2. 通过调用random_elements.py中的`get_attack()`生成攻击数据，其中会对`stdin_format`文件进行操作，将其转化为列表，再对列表中的每一项进行正则表达式校对，去除掉括号逗号这样的无用信息。得到一个存有类似`["int", 1, 2]`的三元列表的二维列表，最后再通过遍历整个列表时，调用对应的生成随机数函数（同样在类中），生成一个二维列表。而使用者可以通过改变`attack_times`参数，修改生成的攻击次数。
3. 遍历输入文件夹中的每一个文件，通过调用runner.py中的`get_answer()`得到其运行attack的结果，在该函数中，首先会调用`subprocess.run`利用shell执行g++编译c++文件，编译后再执行生成的可执行文件，利用管道导入输入文件，并且将`std::cout`的内容传入`result.txt`文件，再将其以字符串的形式读出来，将每一次的攻击的结果整合成一个列表。
4. 调用outputter.py中的`init_output()`，生成储存等价文件和不等价文件对的csv文件。遍历每一对文件对，将其的结果列表进行比对，结果列表包含`return_code`和`std::cout`输出内容，若结果相同，则调用outputter.py中的`get_equal_output()`，否则调用`get_inequal_output()`，这两个函数需要的参数是两个文件和对应的文件夹，在函数中对对应的csv文件写入文件对和对应路径即可。完成后将生成的中间文件删除即完成。

由网站给出的样例，最后生成了94对数据。

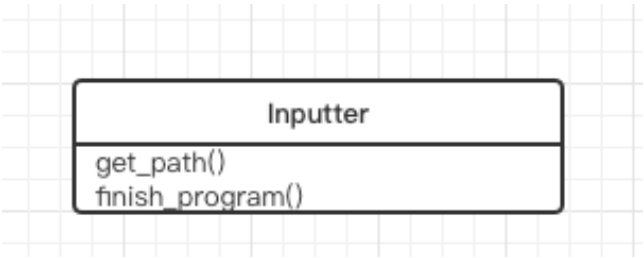


```
lab4 > output > inequal.csv
1 file1,file2
2 input/4A/101036360.cpp,input/4A/117364748.cpp
3 input/4A/101036360.cpp,input/4A/127473352.cpp
4 input/4A/101036360.cpp,input/4A/134841308.cpp
5 input/4A/101036360.cpp,input/4A/173077807.cpp
6 input/4A/101036360.cpp,input/4A/48762087.cpp
7 input/4A/101036360.cpp,input/4A/84822638.cpp
8 input/4A/101036360.cpp,input/4A/84822639.cpp
9 input/4A/117364748.cpp,input/4A/127473352.cpp
10 input/4A/117364748.cpp,input/4A/134841308.cpp
11 input/4A/117364748.cpp,input/4A/173077807.cpp
12 input/4A/117364748.cpp,input/4A/48762087.cpp
13 input/4A/117364748.cpp,input/4A/84822638.cpp
14 input/4A/117364748.cpp,input/4A/84822639.cpp
15 input/4A/127473352.cpp,input/4A/173077807.cpp
16 input/4A/127473352.cpp,input/4A/48762087.cpp
17 input/4A/127473352.cpp,input/4A/84822638.cpp
18 input/4A/127473352.cpp,input/4A/84822639.cpp
19 input/4A/134841308.cpp,input/4A/173077807.cpp
20 input/4A/134841308.cpp,input/4A/48762087.cpp
21 input/4A/134841308.cpp,input/4A/84822638.cpp
22 input/4A/134841308.cpp,input/4A/84822639.cpp
23 input/4A/173077807.cpp,input/4A/48762087.cpp
24 input/4A/48762087.cpp,input/4A/84822638.cpp
25 input/4A/48762087.cpp,input/4A/84822639.cpp
26 input/50A/138805414.cpp,input/50A/164831265.cpp
27 input/50A/138805414.cpp,input/50A/21715601.cpp
28 input/50A/138805414.cpp,input/50A/29019948.cpp
29 input/50A/138805414.cpp,input/50A/31034693.cpp
30 input/50A/138805414.cpp,input/50A/33794240.cpp
31 input/50A/138805414.cpp,input/50A/36641065.cpp
32 input/50A/138805414.cpp,input/50A/45851050.cpp
33 input/50A/142890373.cpp,input/50A/164831265.cpp
34 input/50A/142890373.cpp,input/50A/21715601.cpp
35 input/50A/142890373.cpp,input/50A/29019948.cpp
36 input/50A/142890373.cpp,input/50A/31034693.cpp
37 input/50A/142890373.cpp,input/50A/33794240.cpp
38 input/50A/142890373.cpp,input/50A/36641065.cpp
39 input/50A/142890373.cpp,input/50A/45851050.cpp
40 input/50A/164831265.cpp,input/50A/21508887.cpp
41 input/50A/164831265.cpp,input/50A/21508898.cpp
42 input/50A/164831265.cpp,input/50A/30534178.cpp
43 input/50A/164831265.cpp,input/50A/45851050.cpp
44 input/50A/21508887.cpp,input/50A/21715601.cpp
45 input/50A/21508887.cpp,input/50A/29019948.cpp
46 input/50A/21508887.cpp,input/50A/31034693.cpp
47 input/50A/21508887.cpp,input/50A/33794240.cpp
48 input/50A/21508887.cpp,input/50A/36641065.cpp
```



```
lab4 > output > equal.csv
1 file1,file2
2 input/4A/127473352.cpp,input/4A/134841308.cpp
3 input/4A/173077807.cpp,input/4A/84822638.cpp
4 input/4A/173077807.cpp,input/4A/84822639.cpp
5 input/4A/84822638.cpp,input/4A/84822639.cpp
6 input/50A/138805414.cpp,input/50A/142890373.cpp
7 input/50A/138805414.cpp,input/50A/21508887.cpp
8 input/50A/138805414.cpp,input/50A/21508898.cpp
9 input/50A/138805414.cpp,input/50A/30534178.cpp
10 input/50A/142890373.cpp,input/50A/21508887.cpp
11 input/50A/142890373.cpp,input/50A/21508898.cpp
12 input/50A/142890373.cpp,input/50A/30534178.cpp
13 input/50A/164831265.cpp,input/50A/21715601.cpp
14 input/50A/164831265.cpp,input/50A/29019948.cpp
15 input/50A/164831265.cpp,input/50A/31034693.cpp
16 input/50A/164831265.cpp,input/50A/33794240.cpp
17 input/50A/164831265.cpp,input/50A/36641065.cpp
18 input/50A/21508887.cpp,input/50A/21508898.cpp
19 input/50A/21508887.cpp,input/50A/30534178.cpp
20 input/50A/21508898.cpp,input/50A/30534178.cpp
21 input/50A/21715601.cpp,input/50A/29019948.cpp
22 input/50A/21715601.cpp,input/50A/31034693.cpp
23 input/50A/21715601.cpp,input/50A/33794240.cpp
24 input/50A/21715601.cpp,input/50A/36641065.cpp
25 input/50A/29019948.cpp,input/50A/31034693.cpp
26 input/50A/29019948.cpp,input/50A/33794240.cpp
27 input/50A/29019948.cpp,input/50A/36641065.cpp
28 input/50A/31034693.cpp,input/50A/33794240.cpp
29 input/50A/31034693.cpp,input/50A/36641065.cpp
30 input/50A/33794240.cpp,input/50A/36641065.cpp
31
```

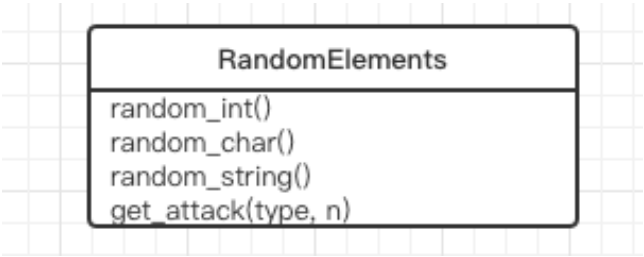
处理输入inputter



Inputter类中存有两个函数，第一个是`get_path()`，用于得到输入的路径，虽然在本次实验中路径的设置较为简单，但是

将其封装起来有利于拓展其功能。第二个函数式`finish_program()`是在结束输入后，将生成的一些txt文件删除掉。

随机生成攻击输入random_elements



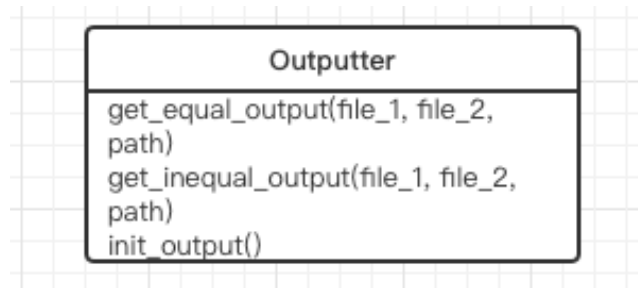
首先是随机生成三种数据，int直接调用了python的random库中的randint，char则是先调用string.ascii_letters得到一个含有所有大小写字母的列表，然后用random.choice()选择其一即可，string则多次调用random_char()即可。在get_attack()中，利用给定的输入格式列表和攻击次数，先通过循环得到一次攻击的attack_row，再将n个attack_row列表连接成一个二维列表attack_list返回。

运行程序runner



Runner中get_answer()的主要功能是编译文件并执行文件，将由attack_list输入得到的结果存在二维列表中返回。为了编译并运行可执行文件，调用了subprocess.run()，且将shell设置为True，即可输出指令，用g++编译后，利用管道将输入文件传入，并将std::cout中的内容导入到result.txt文件，再将其转化为字符串，并与subprocess.run().runcode()（程序返回值）一起形成二维二元列表。

处理输出outputter



在输出列表得到后，将会开始进行比对和输出，比对前先会对两个输出的csv文件进行初始化（创建并且把第一行写入），judger比对完后，将会分情况调用`get_equal_output()`和`get_inequal_output()`，在这两个函数中，通过输入得到文件及路径，会把路径和文件名打包好，根据格式写入csv文件。

git相关

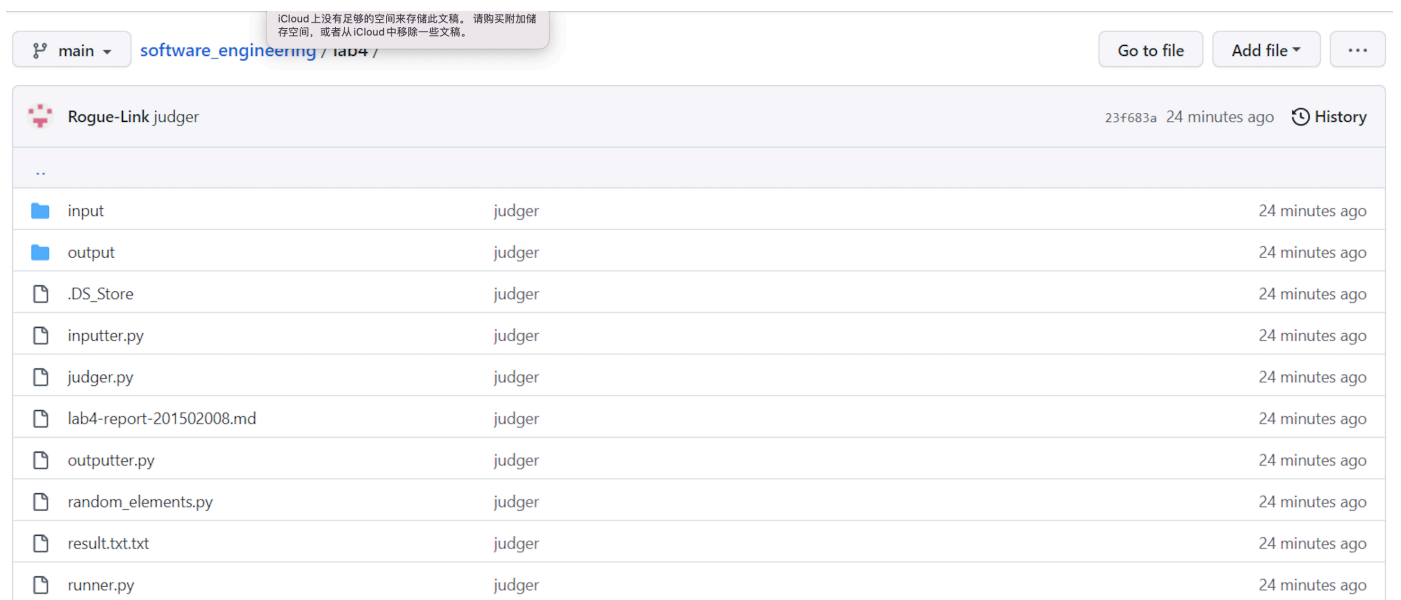
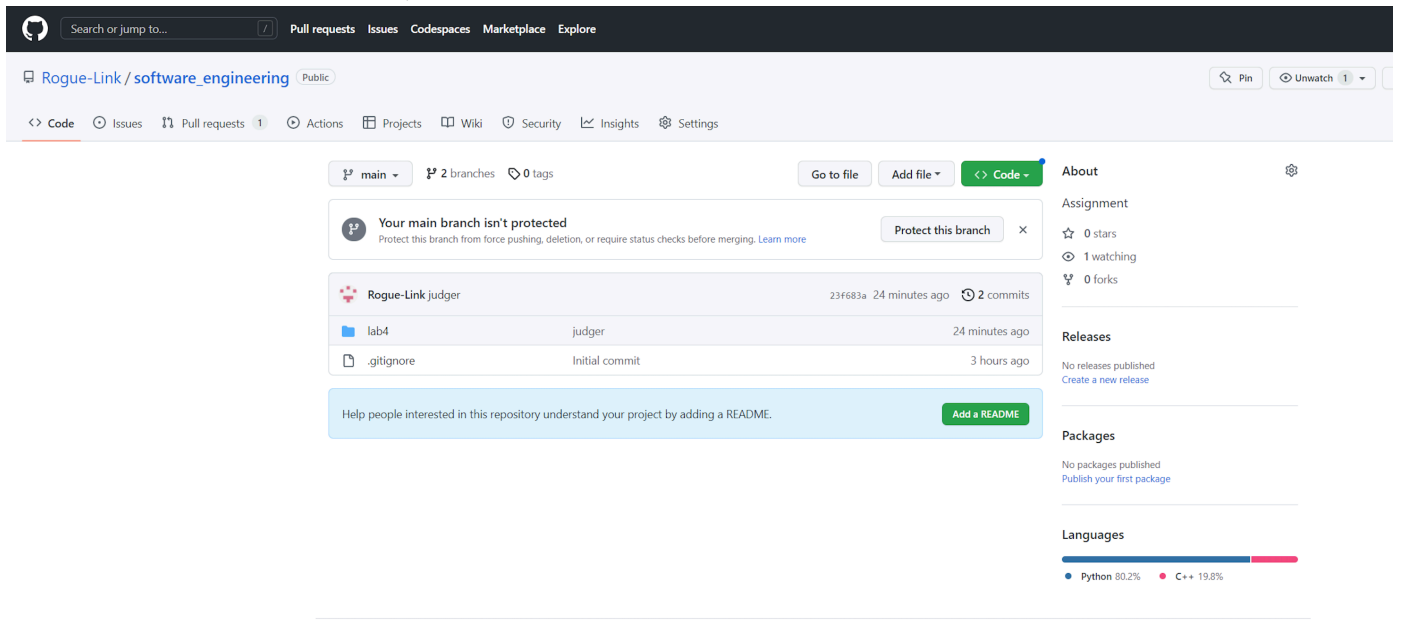
```
$ git log --graph --decorate --oneline --simplify-by-decoration --all
* 23f683a (HEAD -> main) judger
| * 2f2ad68 (origin/judger) Add files via upload
|/
* bfa2e8c (origin/main, origin/HEAD, runner, random_elements, outputter, judger, inputter) Initial commit
```

在写代码期间，将五个部分分别建立分支分别工作，当完成该部分时，将修改文件git add并commit后即可。在完成后，使用git rebase进行合并历史修改（后面会解释为什么不用git merge），因为期间没有遇到过需要回退的重大问题，并没有使用过git reset。

在完成代码工作后，在github上新建了仓库：https://github.com/Rogue-Link/software_engineering/tree/main

```
84685@DESKTOP-70585Q9 MINGW64 /c/垃圾/jiawang/software_engineering (main)
$ git push
Enumerating objects: 38, done.
Counting objects: 100% (38/38), done.
Delta compression using up to 8 threads
Compressing objects: 100% (33/33), done.
Writing objects: 100% (37/37), 6.77 KiB | 1.69 MiB/s, done.
Total 37 (delta 3), reused 1 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/Rogue-Link/software_engineering.git
bfa2e8c..23f683a main -> main
```

并且在仓库中的main分支下，放入了所有源码和输入的样例以及对应的输出。



亮点与挑战（一些经验）

在删除每一个文件时，要先判断这个文件是否存在，如果存在可能不存在文件的情况，要进行判断，而本次程序中所有的删除都进行了判断。

Mac在调用gcc和g++时会强行调用clang，就算更改路径也无法解决，而clang无法对一些比较偏僻的语法进行编译，比如没有int在前的main()函数，因此在跑样例时使用了虚拟机。

```
return_code = subprocess.run("cat " + path + "/" + "put.txt" + "|" + "./a.out > result.txt", shell = True)
```

在上述代码中使用了管道进行传入输入文件，由于本专业没有操作系统，所以自学了管道相关知识。

在最后进行代码历史更改记录合并时，并没有选择git merge，而是使用了git rebase，因为merge会把公共分支和你当前的commit 合并在一起，形成一个新的 commit 提交，而rebase会把当前分支的 commit 放到公共分支的最后面，相比于前者，后者更能维护历史记录，有着更好的代码更新结构。

在处理输入形式的时候，调用了正则表达式re库，通过比对去除了无用符号并转换为了列表。

给助教老师的提醒

因为用的是MacOs系统，所以不一定能在所有系统上正确运行（小概率事件），老师在运行的时候要是遇到问题请联系qq：846853688