基础部分（Wordle）

1. 程序结构。

学生使用了助教提供的Wordset类，用于检测输入的答案和猜测单词是否在列表中。然后，创建了一个类用于储存单词在猜测后输出的GYR信息，以及整个字母表中的GYRX信息；在这个类中，Check函数用于读取输入的单词并输出该单词对应的GYR信息；CheckUse函数用于检查单词是否满足困难模式要求；Clear可以清空数据，用于重新开始一轮游戏。

TestFin与TestAcc的设计比较冗余，起初设计这两个函数是因为学生不太理解助教所提供的函数，后来就一直保留下来。

WordIn函数用于承载游戏主体，ans输入代表游戏答案，mod输入用于选择普通或困难模式，result可修改，用于表示游戏成功或失败，times用于记录游戏局数，后两个是为task4提供的特殊变量。

之后的四个Task分别执行的是4个需求。

主体的算法是：先判断单词是否标为G，如果是，把它的位置标为used防止后续再用到，然后判断Y，如果前两者都不是，则标为R。把这些信息都记录在wordlist上，用于输出。

1. 主要功能。

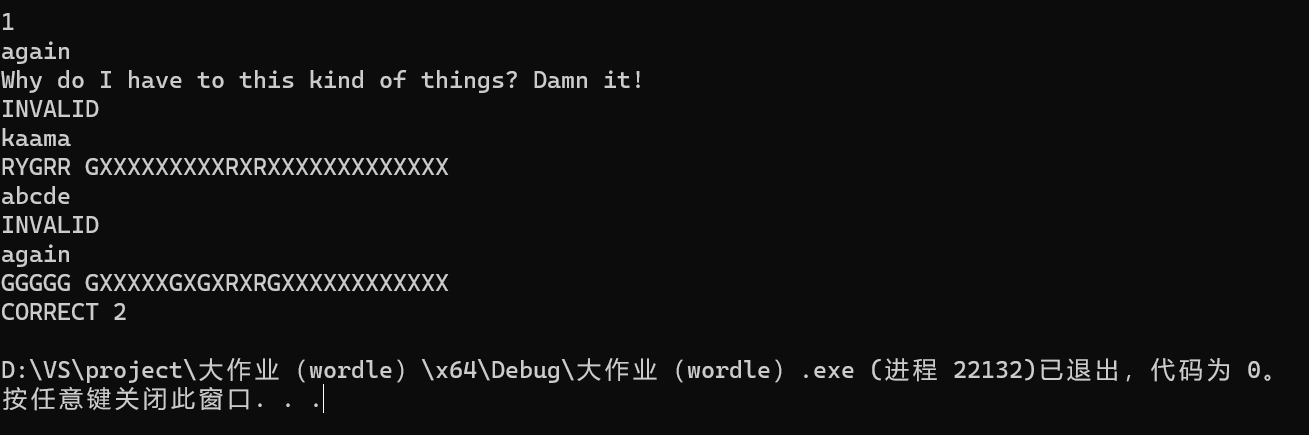
先输入一个1-4的数字，用于选择模式。

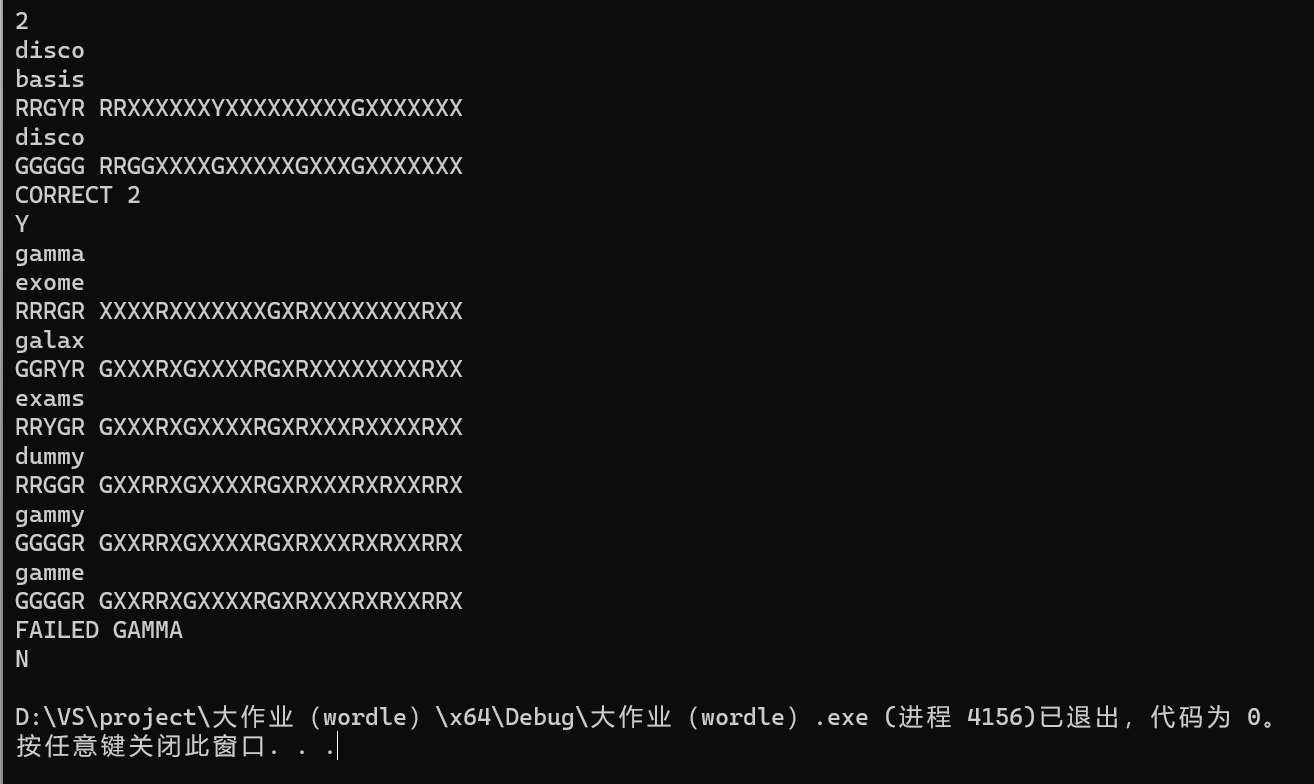
若为模式1，输入答案，系统会检测答案是否属于“FINAL.txt”文件，如果不是，则输出：INVALID，并需要重新输入，直到满足条件。然后，就可以输入猜测的单词，如果不属于“ACX.txt”文件，同样会输出：INVALID。如果满足，就会输出对该单词的GYR判断，以及整体单词表的GYRX判断。空的输入会被自动忽略。

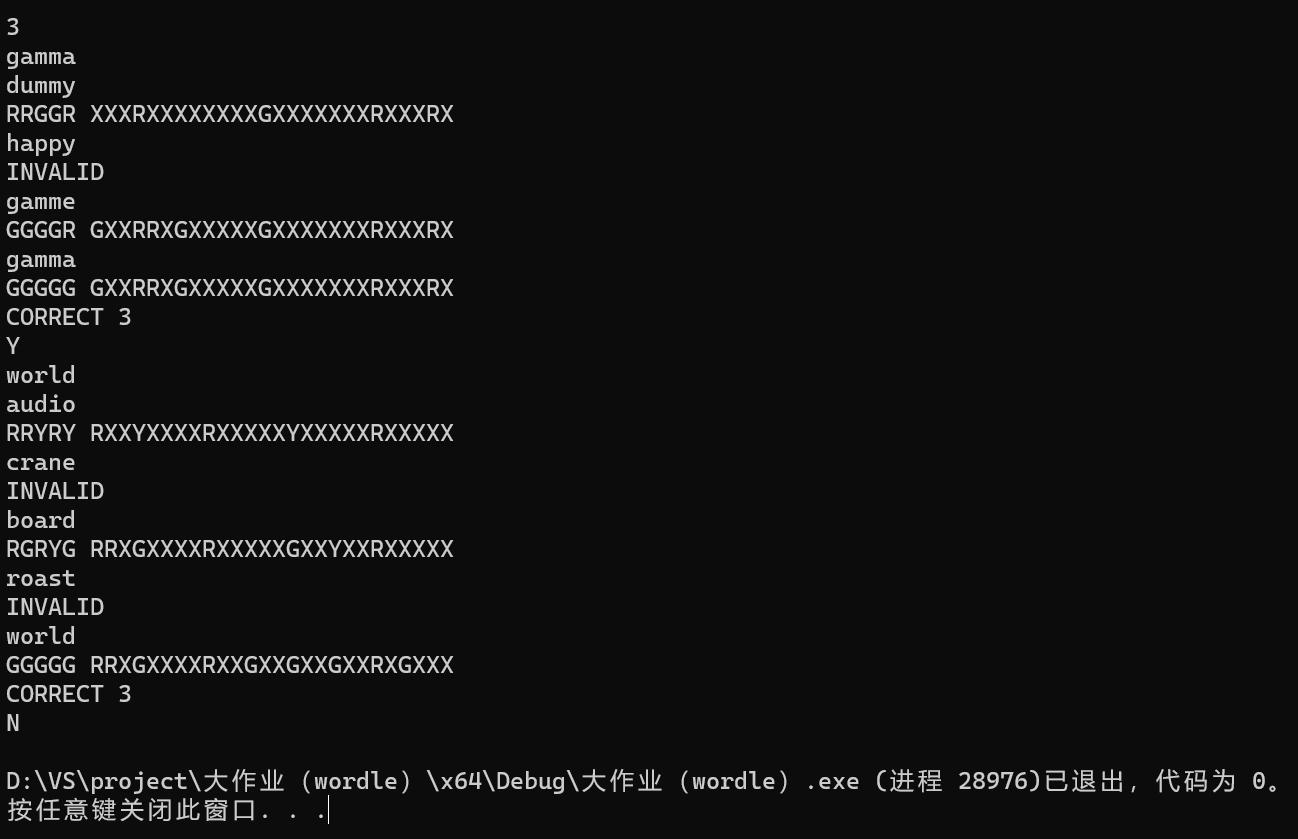
模式2就是在模式1的基础上添加多轮游戏，输入Y则开始新游戏，输入其它，则游戏停止。

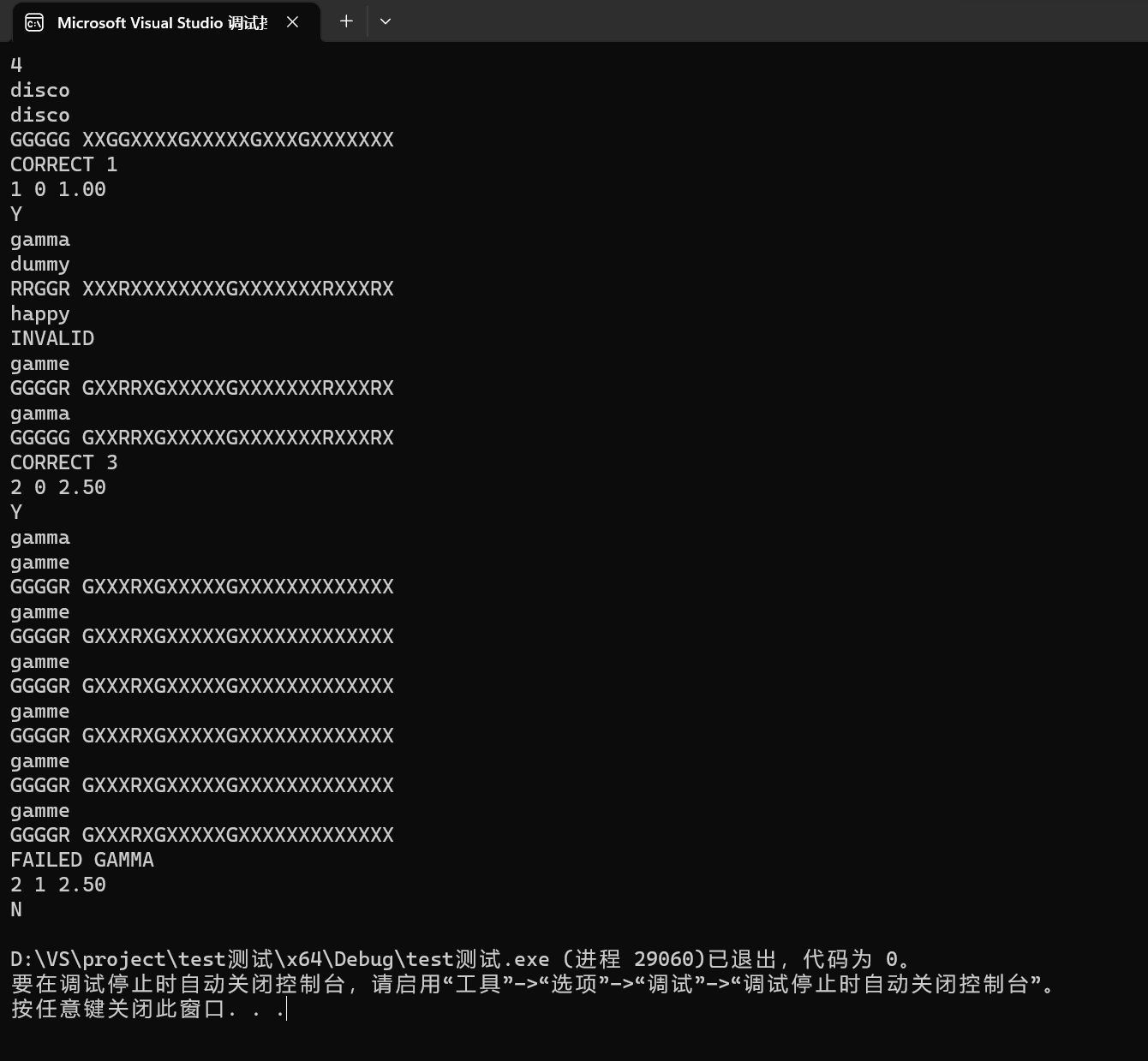
模式3增加困难模式检测，如果输入不满足困难模式的要求，同样会输出：INVALID。

模式4无多余操作，只是会记录平均每个胜利局的猜测次数。









提高部分（Wordle求解器）

1. 实现方式。

学生借鉴了困难模式的算法。首先，把ACC.txt的单词输入进vector<string>容器，然后，创建一个类用来保存每个单词的信息熵。在获取了wordle给出的结果后，根据wordle的GYR判断，剔除掉容器中不符合下列条件的单词：

原本单词结果是‘G’的，在相同的位置上要有相同的字母。

原本单词结果是‘Y’的，在其他非‘G’的位置要出现这个字母，且这个字母不应该出现在Y的位置上。

原本单词结果是R的，不应该在任何位置出现这个字母。

之后，按照信息熵越大越优先的原则推荐单词。

1. 主要功能。

首先，输入数字用于选择模式。

输入1，会进入普通模式，系统会推荐5个比较合适的单词，玩家需输入自己猜测的单词以及wordle给出的结论，系统经过上述算法后会再推荐5个单词，以此类推。

输入2，会进入交互模式，系统会推荐1个单词，玩家需在wordle上输入系统推荐的单词，并告知系统wordle给出的结论，系统会再次推荐一个单词，以此类推。

1. 实现问题。

在完成找出最优起始词的需求时，我应用了以下思路：

在ACC中遍历所有词作为起始词，找出对每个FINAL中单词所对应的猜测次数然后求和，作为该单词的总猜测次数，猜测方法为求解器的2模式。

但是在实现过程中，系统总是给出abort() has been called的问题，这似乎是因为我的程序运行时间过长所致，我在尝试修改之后没有什么改变，这导致我最终没有找出最优起始词的文档，只实现了求解器部分。

1. 感想。

在程序设计中，语言只是一种工具，真正重要的是背后的数学算法，编程的精髓还是在数学。