评分标准

1. 阅读、理解代码 (30%)

2. 功能完成度 (70%)

1. 游戏的创建和销毁 (20%)
    - (10%) 读取与显示：对于给出的每一张示例地图，你的程序都成功完成了 `Game` 对象的初始化，并将游戏地图正确显示出来。显示结果不应该出现乱码。特别地，在下面这张包含了一些特殊情况的地图上（`maps/special.txt`），你的程序应该能正常工作。

      ```
      7 18 2
        RRRRRRRRRRRRR
        R...........R
      RRR 0  BBBB    RRR
      R......B1 B......R
      RRR    BBBB  C RRR
        R...........R
        RRRRRRRRRRRRR
      up
      left
      ```

      当然，地图的格式是允许修改的。
    - (5%) 没有内存泄漏。注意， `destroyGame(pGame)` 必须能释放 `pGame` 所持有的一切资源。
    - (5%) 你在合适的时机关闭了地图文件。
2. 鬼和 Pacman 的移动 (20%)
    - (7%) 鬼和 Pacman 应该按照你设定的规则移动，并且被正确更新到屏幕上（特别是 Pacman 的颜色也要正确）。
    - (7%) 鬼出现重叠时，你的游戏应该按照期望运作。你需要解释你是如何做到的。当然，也有可能你根本无需考虑这个问题，如果是这样的话请解释为什么。
    - (6%) 移动应当快速地完成，比如，不应遍历 `grid` ，不应通过清屏+重新输出整个游戏画面的方式来更新。
3. Pacman 死亡 (10%)
    - 当 Pacman 与鬼重叠时， Pacman 应该死亡。这个对于死亡的判断应当非常快，甚至不应遍历所有的鬼。
4. 关卡相关的功能 (20%)
    - 你实现了什么功能？和 demo 一样？和用户交互选择关卡？支持什么特别的命令行参数吗？解释你实现的东西。

3. 代码质量 (在总得分的基础上，扣除不满足的项对应的分数)

1. (50%) 格式化
2. (50%) 在 GCC13 `-Wall -Wpedantic -Wextra` 之下没有 warning 。
3. (10%) 对函数、变量、类型、预处理宏进行合理的命名。特别地，

    - `i`, `j` 这样的循环变量名可以允许。
    - 一些具有重要含义的变量（特别是某些布尔变量）不应该用 `check`, `flag`, `a`, `b`, `c` 之类毫无意义的名字。
    - 所有类型、函数、函数的参数、全局变量、预处理宏都应该具有有意义的名字，并且遵循一定的命名规范。
4. (10%) 变量应在即将被使用的时候才被声明，并且尽可能不使用全局变量。事实上，我们的 demo 程序完全没有使用全局变量。如果你只想在某两三个函数之间传递某个信息，你应该充分利用参数和返回值。如果某个信息在很多个函数中都要使用，*也许*它适合被存储在一个全局变量里。
5. (10%) 为结构体的多个成员进行初始化，或对所有成员进行逐个赋值时，应尽可能使用 initializer-list （包括 designators） 或者 compound literals ，而不是罗列赋值语句。
6. (10%) 正确维护注释。注释应该被用来解释代码，而不是描述题目。一些作为提示或题目描述的注释应该被删除。 `TODO` 标记的注释在相关功能未实现时应该被保留，实现后应该被改为 `DONE` 或删去。
7. (10%) 正确使用 `enum` ，使用 enum items 来避免 magic numbers 。
8. (10%) 避免重复，包括
    - 充分利用 `moveOneStep`, `oppositeDirection`, `isWall` 等函数，不重新实现这些函数已经实现的功能。
    - 代码中没有大段的雷同，没有不必要的对于分支情况的列举。
9. (10%) 函数的分工明确，特别是 `moveGhosts`, `movePacman`, `pacmanDies` 等函数。