在本课程的作业中，同学们可能遇到的代码错误类型主要为以下几种

1、C++语法错误。

2、运行过程中的段错误。

3、算法逻辑错误。

4、计算时效性的问题。

针对问题一：

这类错误一般编译回报错，并且 IDE 会有表示，建议大家多多学习 C++的语法基础，大多数问题即可解决。倘若无法直接确认错误的解决方法，可以将报错信息在 google 或 bing 上检索，大多数也可以解决。

针对问题二：

本作业中可能会出现数组越界、计算不合法（除数为 0.0）等错误。这类问题的核心点在于定位报错位置。

先在此提出一种易上手的调试方法，即运用 C++的标准输出或者 ROS 的日志系统，如 ROS_INFO()等工具，将所需信息打印至终端屏幕上。

当你理解代码的整体框架，并理解这个程序的运行顺序时，可以分析出可能出错的位置，在其前后打印对应信息，便可逐步定位出错信息，倘若不太理解整个程序的运行顺序，则建议在你修改过的代码位置前后，打印对应信息，利用二分法的思路逐步定位信息。

当然，段错误的可能原因还有内存越界、访问不存在的地址、访问空指针（对象未实例化）、内存溢出等原因，但在本作业中基本不会遇到，有处理其他问代码问题的同学可以自行检索学习。

针对问题三：

这里则需要大家理清楚代码框架，理解清楚算法各个公式的实现位置以及其对应的输入输出，当最终输出错误时，需要从车辆状态输入开始，到最终算法输出结束，推理中间的计算过程，观察其中的数值，比较其与理论推导是否一致，并以此定位问题，分析是哪一步出现计算错误。

以之前有同学提出的“显示车辆速度信息为 0，车辆不动”问题来举例。

首先需要了解整个代码的数据流框架。可以考虑使用 rqt 工具，观察各个节点话题的发布订阅关系，即可理解本代码主要是订阅 ros_bridge 传出的车辆位姿信息，经过算法处理后输出车辆的控制信号，并发送至对应的节点。此时可以先确认两个问题，节点订阅到的位姿信息是否正确传入，最终的控制信号是否正确传出，在前后打印两个信息。当然，倘若位姿信息有问题或者控制信号正常输出但车不动，则可以判断 carla 的服务端出现问题，车辆可能没有正常生成，即可去解决生成车辆节点或者服务端的问题。若这两个地方没有问题，是最终输出的控制信号有问题，则可以再往中间查询，逐步定位是在哪个位置数据计算出现了问题，即可修改。这个方法需要大家多多理解代码运行框架，才能更快的定位问题。

针对问题四：

这是当时我自己遇到的一个问题，lattice_planner 计算超时，造成轨迹输出延迟，可以考虑自行优化算法复杂度（或简化）。定位方法则可以考虑运用 clock()函数计算函数的运行时间与仿真时间做比较，以观察是否超时。

以上方法，应该可以解决大家在课程中可能遇到的代码错误。

学有余力的同学，也可以学习一下 GDB 调试。其可以更快的定位问题，但相对来说比较难上手。在此以我最近在调试的代码举一个例子。

下图是一个内存越界的错误，其不会在函数运行过程中报错，它会在函数运行结束后报错，上述方法一般定位不到，那么需要应用 gdb 调试工具，查看生产的 core 文件，如下，即可发现其具体在代码的那个文件的哪一行产生的错误，即可定位错误，再分析错误原因。除此之外，gdb 调试工具也可以在程序中打断点以方便调试，在此推荐大家一个课程，为牛客网的开源课程，其中有讲解 gdb 调试的视频，有兴趣的同学可以学一学。https://www.nowcoder.com/study/live/504 。还有一个 ROS 写运用 GDB 工具的博客 https://blog.csdn.net/lovely_yoshino/article/details/106882721，当然，这只是一个基础用法，更多功能有兴趣的同学可以依据这些再检索“GDB 调试”学习。