

# FSE598 项目 1

## (作业 1: 50 分和作业 2: 50 分)

计算思维与 VIPLE 编程

### 引言

此项目的目的是确保您已阅读课程课件并理解课件中涵盖的概念，包括计算机组织、计算思维概念、VIPLE 基础知识和 VIPLE 背后的编程概念。当您完成项目后，您应该对这些概念有一个很好的理解，并已将这些概念应用于开发可运行的程序。作业是个人作业。不允许协作完成。所有学生都必须提交独立开发的应用程序。不能重复教材或课程幻灯片中给定的示例。可以遵循这些示例，但必须进行重大更改。

### 实践练习和准备（这部分无需提交）

1. 在 <https://venus.sod.asu.edu/VIPLE/IntroductionVIPLE.pdf>（英文版）上阅读课程和 VIPLE 教程。也可以阅读中文版本：<https://venus.sod.asu.edu/webhome/book/IntroCS2edindex.html>
2. 在 VIPLE Web2D 模拟器中编写第一个迷宫程序，了解迷宫导航算法：  
<https://venus.sod.asu.edu/VIPLE/Web2DSimulator/indexcn.html>
3. 可以从这里开始实现算法：

### Implement Your Algorithm Here

Add a New Line 添加一行

Remove a Line 移去一行

Default: 

Forward 前行

1. If 

Right

 sensor 

>

100

 pixels

Then: 

Turn Right

2. Else if 

Forward

 sensor 

<=

10

 pixels

Then: 

Turn Left

Run 运行

此代码实现什么样的算法？

左沿墙算法？

右沿墙算法？

一个距离贪婪算法？

局部最优？


4. 下载并安装 VIPLE: <https://venus.sod.asu.edu/VIPLE/>

VIPLE 仅在 Windows 计算机上运行。如果您没有 Windows 计算机，您可以在 Mac 计算机上安装 Virtual Box 或在云计算环境上安装虚拟 PC。您可以从 Microsoft 网站找到说明，例如：<https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/>

现在，您可以从以下位置下载 VIPLE：<https://venus.sod.asu.edu/VIPLE/>

---

- **ASU VIPLE Software and Edison Middleware Free Downloads**

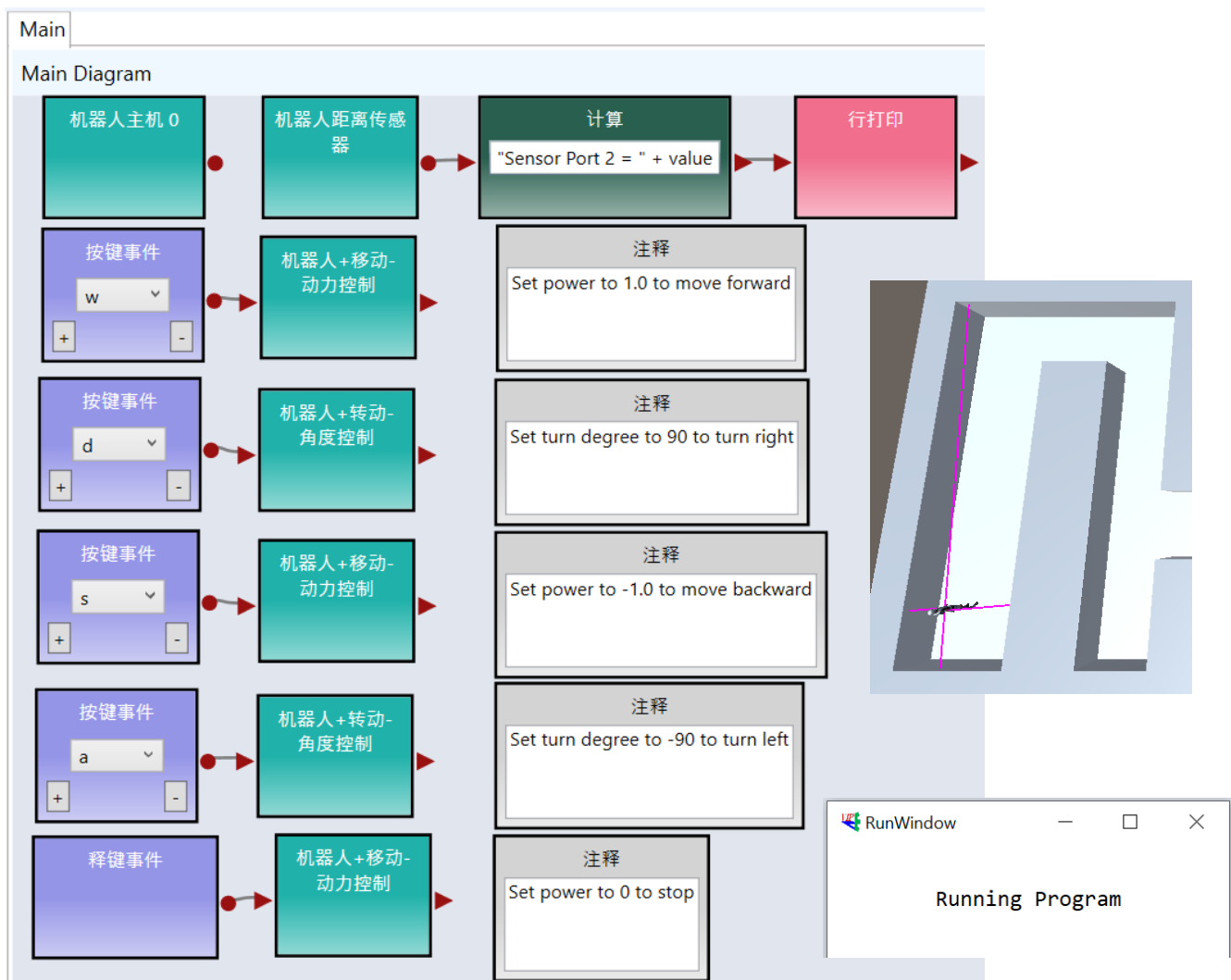
-  [ASU VIPLE Standard Edition Installer](#) the latest version for Windows 10. you are not sure which one you need.
- [ASU VIPLE Downloadable Version](#) for Windows 10. This edition contains a application from file: VIPLE, which is the largest executable in the folder.
- [ASU VIPLE Downloadable Version](#) for Windows 7. This edition contains a v

安装版本位于：

<https://venus.sod.asu.edu/VIPLE/Install/publish.htm>

它可能需要运行 ASU VPN 才能运行安装程序。在这种情况下，您可以在以下位置使用下载版本：<https://venus.sod.asu.edu/VIPLE/Release.zip>

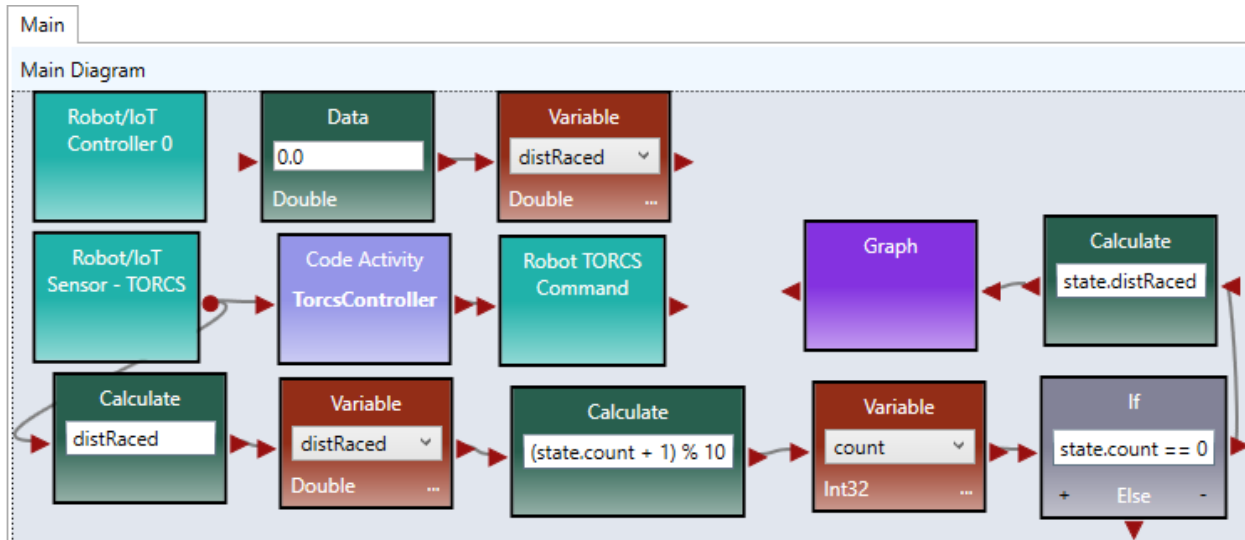
5. 按照教程，使用 Unity 模拟器 2 运行线控程序。请注意，若要启动程序，您需要：
  - 1) 开始运行“Unity 模拟器 2”后，您需要返回到 VIPLE，单击“开始”按钮（绿色箭头）。然后，将显示“控制”窗口，显示机器人正在运行。
  - 2) 当使用键盘控制机器人时，必须将控制窗口 (RunWindow) 以活动模式保持在顶部，而不是后台。



## 6. 采用 C# 的基本赛车实现

本练习的目的是对 TORCS 赛车进行编程，使其能够在赛道中自动驾驶。目标是用最少的时间完成赛道。必须阅读 TORCS 网站 (<http://torcs.sourceforge.net/>) 和课程 PPT，以了解详细信息，并编写比给定代码更高效的代码。

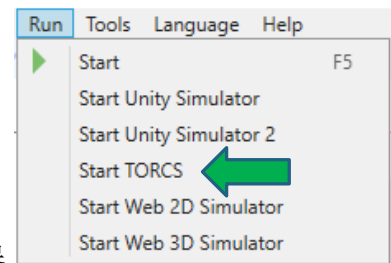
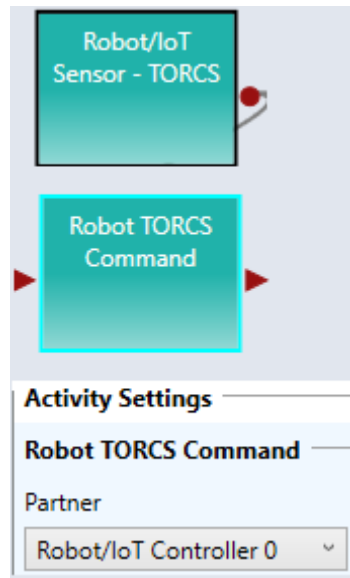
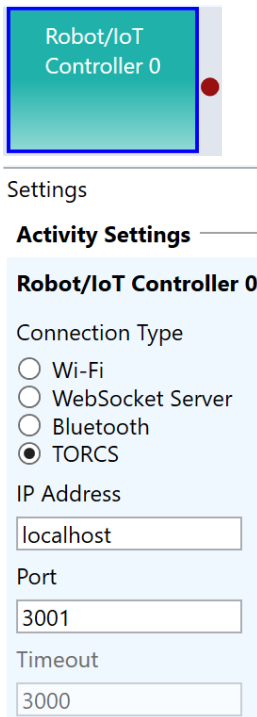
步骤 1. 实现以下 VIPLE 代码：



对于代码活动，可以输入以下代码：

```
using System;
using System.Collections.Generic;
[Serializable]
public class TorcsController : CodeUtilities.CodeBase
{
    public TorcsController()
    { } // 若要执行代码，必须重写 Execute 方法。
    public override void Execute()
    {
        // 值将作为对象传入，需要将其拆解。
        var inputDictionary = Input as Dictionary<string, object>; // 返回一个字典，允许我们按名称引用值。
        var toReturn = new Dictionary<string, double>()
        {
            {"accel", 0.3},
            {"brake", 0.0},
            {"gear", 2},
            // 因为传入的值为对象，需要将这些值转换为浮点型。
            {"steer", (double)inputDictionary["angle"] - (double)inputDictionary["trackPos"] * 0.5},
            {"clutch", 0.2}
        };
        Output = toReturn;
    }
}
```

步骤 2.按如下所示配置控制器和设备。



步骤 3：启动模拟器

当首次在 VIPLE 中启动 TORCS 时，它应该将您链接到 TORCS 下载站点。如果没有，您可以转到 TORCS 官方站点来下载它，或者直接转到：

<http://torcs.sourceforge.net/index.php?name=Sections&op=viewarticle&artid=3#linux-src-all-win>

将 TORCS 安装到 VIPLE 目录，路径为：Documents\VIPLE\_Files\TORCS\wtorcs.exe

步骤 4.选择“Race”

运行 TORCS 模拟器后，选择：Race



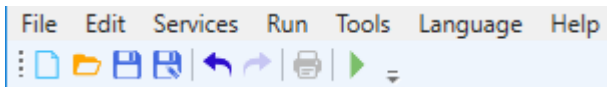
步骤 5.选择“Basic Quick Race”



步骤 6.选择“New Race”



步骤 7.现在，通过单击绿色三角形，启动 VIPLE 程序：



赛车应立即启动。注视比赛，并回答以下问题：

赛车是否运行良好？完成赛道需要多少时间？如果是您在驾驶赛车，能缩短时间吗？

#### 7. 改进赛车代码

双击“代码活动”，如下图所示。

```
Code Editor
1 using System;
2 using System.Collections.Generic;
3
4 [Serializable]
5 public class TorcsController : CodeUtilities.CodeBase
6 {
7     public TorcsController()
8     {
9     }
10
11
12 // To execute your code, you must override the Execute method.
13 public override void Execute()
14 {
15     // The values are passed in as an object, need to unbox it.
16     var inputDictionary = Input as Dictionary<string,object>;
17
18     // Returning a dictionary allows us to reference the values by name.
19     var toReturn = new Dictionary<string, double>()
20     {
21         {"accel", 0.3},
22         {"brake", 0.0},
23         {"gear", 2},
24         // Need to cast these values to doubles, as they are passed in as objects.
25         {"steer", (double)inputDictionary["angle"] - (double)inputDictionary["trackPos"] * 0.5},
26         {"clutch", 0.2}
27     };
28
29     Output = toReturn;
30 }
31 }
```

比赛由参数确定，包括加速(accel)、制动(brake)、档位(gear)、转向(steer)、角度(angle)、赛道位置(trackPos)和离合器(clutch)。尝试通过试错法修改这些参数值，以提高比赛成绩。

请注意，所有参数都使用常数值，包括加速、制动、档位和离合器。若要提高比赛成绩，您可能想要在不同的情况下给定不同的值。例如，在直道加大加速值，在弯道增加制动值，等等。

## 8. 采用 Python 的赛车实现

用 Python 代码活动来实现。

## 项目作业：高级 VIPLE 编程（这部分需要提交）

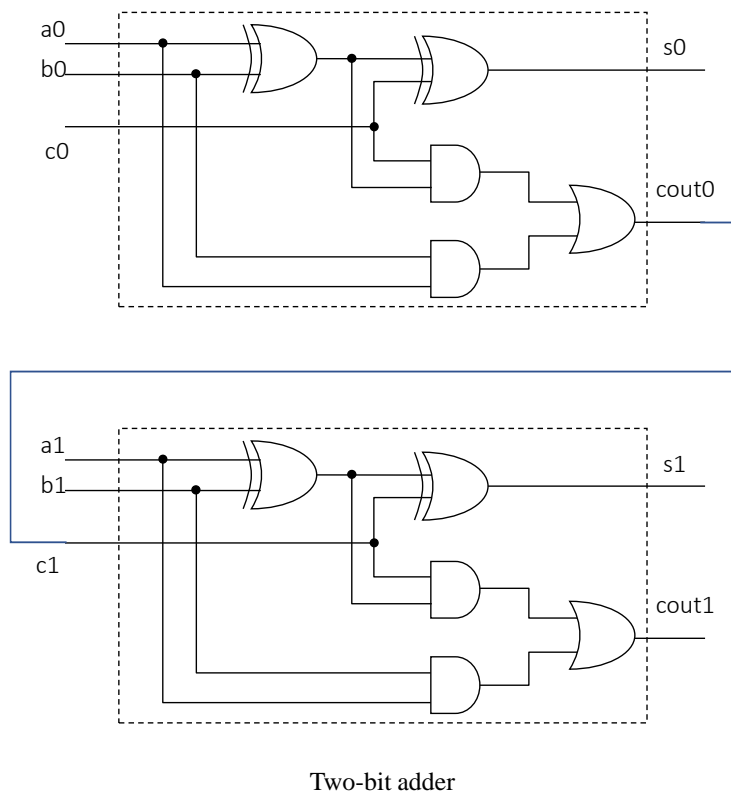
本项目有两个作业

### 作业 1（50 分）

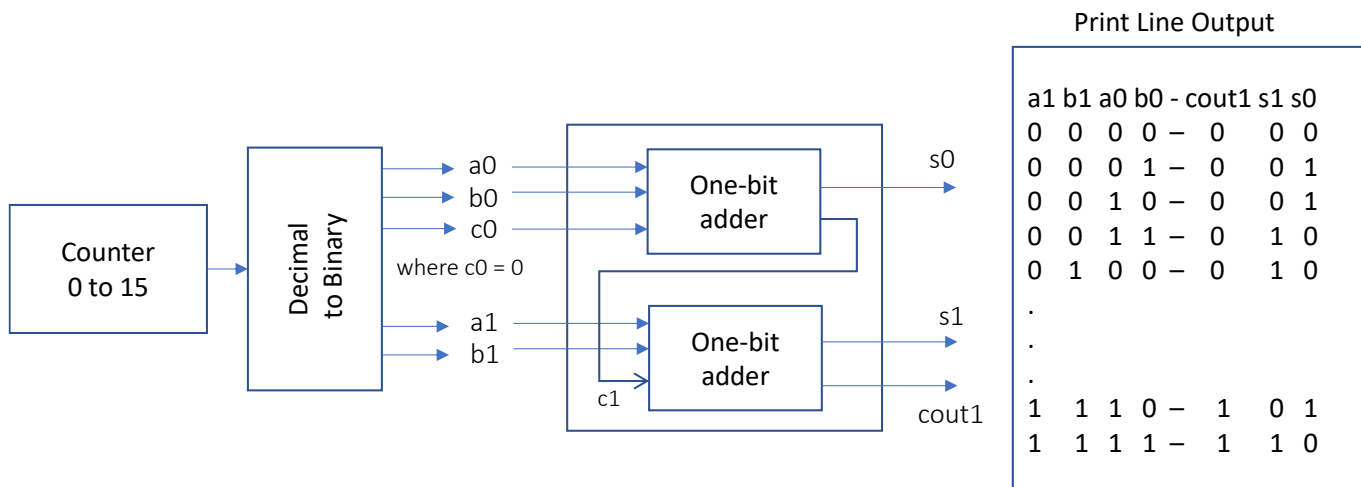
在开始这部分作业之前，您需要按照课程幻灯片和本文档的实践练习部分作准备，以开始在 VIPLE 工作流中编程。

问题 1 下图给定 两位加法器的电路：

[25 分]



- 1.1 使用自定义活动实现每种逻辑门。 [5]
- 1.2 使用自定义活动实现一位加法器。然后，使用一位加法器作为组件，在 **VIPLE Main** 活动中 (或在一个自定义活动中) 实现两位加法器。 [10]
- 1.3 自动化测试：使用一个计数器(Counter) 自定义活动（模块）来生成数字 0, 1, 2, 3, 4, ..., 14, 15, 并用另一个模块将十进制数字转换为二进制数字以作为两位加法器的输入。 [10]





问题 2 Unity 模拟器 2 的机器人迷宫导航。

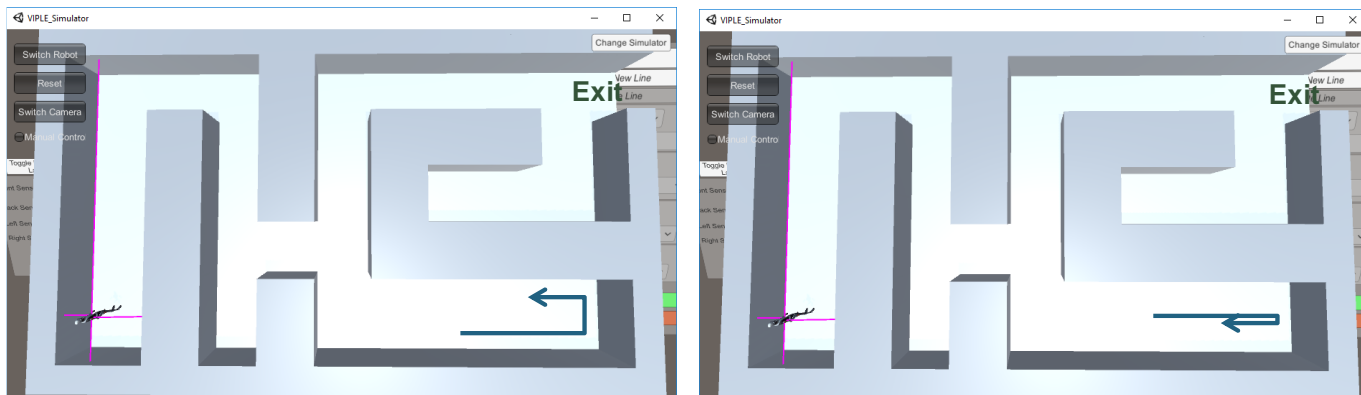
[25 分]

2.1 按照课程课件实现基本的右沿墙程序。您可能需要调整参数值。

[15]

2.2 实现改进的（修改的）右沿墙程序，该程序应使用较少的转弯或缩短移动到出口点的距离。例如，可以检测到 **cul-de-sac**（死胡同）。在给定的右沿墙程序中，机器人在死胡同中将先转 90 度，接着向前行进，然后再转 90 度。在增强的程序中，您可以使用更多传感器，以便在检测到两侧都没有路可走时，使机器人进行 180 度转弯，如下图所示。

[10]



您也可以进行其他方面的改进，只要您的算法可以转更少的角度。请注意，仅将程序改为左沿墙算法是不可以的，这不会增强性能。它能增强机器人在此迷宫的性能，但在其他不同的迷宫中性能可能会更差。

作业 1 提交：Adder2.viple 包括本第一题中的所有代码。提交：2.1 的代码和 2.2 的代码，包括以下文档和代码文件：

- Adder2.viple
- Maze1.viple
- Maze2.viple

将所有问题中的所有代码放入一个文件夹中，压缩该文件夹，然后提交 zip 文件。

## 作业 2（50 分）

问题 3. 根据本文档第一个部分中的实践练习，按照模块 1 单元 2 课程 5 实现一个高级赛车程序。

[50 分]

3.1 提交完整的 VIPLE 程序（可包括 C# 和/或 Python 程序作为代码活动）。确保程序可在其他机器上直接运行。

[10 points]

提交高级赛车程序 [10 points]

提交简单/基础赛车程序 [5 points]

3.2 提交包括赛车运行时间（以秒为单位的时间）的屏幕截图。提交运行时间仅供参考。TA 将使用相同的计算机和配置来测试运行时间。

评分标准：此项目问题是一个竞赛，将根据所有人的程序的相对性能表现进行评分。

- 在提交的程序中，排名前 10%（使用最少的时间完成比赛）：100% = 40 分。
- 在提交的程序中，排名下一个 10%（前 11% – 20%）：95% = 38 分。
- 在提交的程序中，排名下一个 10%（前 21% – 30%）：90% = 36 分。
- 在提交的程序中，排名下一个 10%（前 31% – 40%）：85% = 34 分。
- 在提交的程序中，排名下一个 10%（前 41% – 50%）：80% = 32 分。
- 在提交的程序中，排名下一个 10%（前 51% – 60%）：75% = 30 分。
- 在提交的程序中，排名下一个 10%（前 61% – 70%）：72.5% = 29 分。
- 在提交的程序中，排名下一个 10%（前 71% – 80%）：70% = 28 分。
- 在提交的程序中，排名下一个 10%（前 81% – 90%）：67.5% = 27 分。
- 在提交的程序中，排名下一个 10%（前 91% – 100%）：65% = 26 分。

作业 2 提交：第 3 题中的所有代码，包括以下文档和代码文件：

- Race.vipl 代码文件和屏幕截图

将所有问题中的所有代码放入一个文件夹中，压缩该文件夹，然后提交 zip 文件。

## 提交要求和说明

请确保您整理后提交的文件夹中包含所有的文件，包括解决方案文件、项目文件、和代码。这样您提交的项目就可以直接被打开并测试，而不需要重新组织和整理您的文件。

然后，将您所有要提交的文件复制到一个文件夹中，并将此文件夹压缩，在Canvas界面上上传并提交您的作业。

**提交作业注意事项：**该作业由多个分布式任务和组件组成。当您创建它们时，它们可能存储在您电脑的不同位置。您必须在创建项目时选择一个新的位置来存储这些文件。然后，您可以将这些文件复制到一个文件夹中，以便在Canvas上提交。为了确保您在压缩文件中包含了所有的文件，并且它们能一起工作，您必须在提交前测试它们。您也必须从Canvas上下载您提交后的文件。然后在不同的机器上解压并测试您的作业，看看您是否能在不同的机器上运行解决方案，因为助教会在不同的机器上测试您的应用程序。

如果您提交了一个空的文件夹，或者一个不完整的文件夹，我们将不能对您在截止日期后重新提交的部分进行评分！我们只对您在截止日期前提交的作业进行评分。

## 分数和评判标准

每个小问题（编程任务）都有一定的分数。我们将按照以下步骤对您的程序进行评分：

(1) 编译代码。如果不能编译，将扣除50%的编译代码的分数。然后，我们将阅读代码，并在50%和0之间给分，如下面的评分表的右边部分所示。

(2) 如果代码通过了编译，我们将使用测试案例执行和测试代码。我们将根据评分表的左边部分来给分。

在这两种情况下（编译通过和编译失败），我们将阅读您的程序，并根据分配给每个子问题的分数、您的代码的可读性（代码的组织 and 注释）、逻辑、所需功能的包含以及每个功能的实现的正确性来给分。

请注意，我们不会对您的程序进行调试以弄清错误的大小。您可能会因为一个小错误而失去50%的分数，比如少了一个逗号或一个空格！

我们将对作业中列出的**每个子问题**采用以下评分标准。Pts是用来表示每个子问题的分数。

评分表

评分项	通过编译的代码				代码编译失败		
	pts * 100%	pts * 90%	pts * 80%	pts * 70% -60%	pts * 50% - 40%	pts * 30% -10%	0 or 1
针对于每一个子问题	满足所有要求，注释良好，在所有测试案例中都能正常工作。	在所有测试案例中工作正常。没有提供注释来解释代码的每一部分是做什么的。	可以通过测试但出现一些小问题，如不写注释，代码在某些不常见的边界条件下不工作。	在大多数测试案例中可以工作，但有重大问题，如代码不能通过一个常见的测试。	编译或不能正常工作，但显示出认真努力解决这个问题的态度。	编译失败，显示了一些努力，但代码没有实现所需的工作。	0 分，如果没有提交。  1分，如果提交的文件不能打开、没有内容或没有回答任何问题