**CS 60, Lab 2**

**Classes**

**One person, start a new document in Word or similar.   Include the names of both members of your team and any additional information or code as instructed below. At the end of the lab, please submit your lab report to the Camino submission box with the last names of both members of your group.  Only one submission required per group.  Be sure all group members have a copy of the code.**

**A Mod Class**

1. As you know, the mod operator can be used to find the remainder attained when you divide one number by another. For instance, 9%5 is 4, since the remainder when dividing 9 by 5 is 4.  We often use mod to restrict values to a certain range.  For instance, if you had 180 songs in your Spotify playlist (numbered 0 to 179), and you were playing song 175 and wanted to skip 6 songs, the most sensible thing to do would be to play song 1, since 175+6=181 – There is no song 181, but 181%180=1.

Write a class ModInt to represent numbers in a range from 0 to some fixed number (one less than the mod value).  Your class should have variables for the mod and the numerical value.   For instance, if you wanted to represent that you were playing song 175 on your playlist of 180 songs, you would do so using an object whose value is 175 and whose mod is 180.  You should provide the following functions:

* Two constructors; one that takes no arguments, and one that takes two arguments.
* Getters, but NO setters. If you want to change the value of a ModInt, instead make a new one or use =.

Define all functions besides getters outside the class definition.  Test all functions in your main().  **Copy your code into your lab report**.  **Be sure to get your work approved before moving on.**

1. Add the non-member function:

bool equals(ModInt lhs, ModInt rhs);  Returns true if lhs==rhs and false if not.  lhs’ and rhs’ values and mod must both match.

Test the new function in your main().  **Copy your code into your lab report**.  **Be sure to get your work approved before moving on.**

1. Implement the following member function:

void plusEq(ModInt rhs); modifies the ModInt it is called on to hold the sum of that ModInt and rhs.  You’ll compute the sum by adding the values of the two ModInts, then applying the mod.

To represent the above example of skipping 6 songs when playing song 175 on a playlist of size 180, we would use two ModInt objects, both of which have mod value 180; one (let's call it currentSong) with value 175 and the other (let's call it skipSongs) with value 6; **currentSong.plusEq(skipSongs)**would modify currentSong's value to (175+6)%180 = 1.  Note:  you should only add numbers with the same mod.  If someone attempts to add ModInts with different mods, update the ModInt it’s called on with -1 as both a mod and a value.

**Why must this be a member function?  Put your answer in your lab report.**

Define the method outside the { } for the ModInt class definition.  Test the new function in your main().  **Copy your code into your lab report**.  **Be sure to get your work approved before moving on.**

4.  Implement: ModInt plus(ModInt lhs, ModInt rhs) as a non member function.  Test the new function in your main().  Hint:  the easiest way to do this is to call a method you've already written.  Which one?

**Copy your code into your lab report**.  **Be sure to get your work approved before moving on.**

**-------------------------------------MINIMUM STOPPING POINT---------------------------------------------**

5. Implement non-member function ModInt mult( ModInt lhs, ModInt rhs); Returns a new ModInt that is the product of the two ModInts passed in.   You’ll compute the product by multiplying the values of the two ModInts, then applying the mod.  Note:  as with plus, you should only multiply numbers with the same mod.  If someone attempts to multiply ModInts with different mods, return a ModInt with -1 as both a mod and a value.

Test the new functions in your main().  **Copy your code into your lab report**.  **Be sure to get your work approved before moving on.**

6. Implement non-member functoins lessThan and greaterThan:

* bool lessThan(ModInt lhs, ModInt rhs); Returns true if lhs<rhs and false if not.
* bool greaterThan(ModInt lhs, ModInt rhs); Returns true if lhs>rhs and false if not.

If someone attempts to compare ModInts with different mods, return false.  Test the new functions in your main().  **Copy your code into your lab report**.  **Be sure to get your work approved before moving on.**

**🧩 CS 60 实验 2：类（Classes）**

由一位同学开始新建一个 Word 文档（或类似文档）。  
在文档中写上你们 **小组两位成员的名字**，并在要求处加入相应的信息或代码。  
实验结束后，请将实验报告提交到 **Camino 提交框**，文件名中包含两位成员的姓氏。  
（每组只需提交一次，但请确保每位组员都保存一份代码副本。）

**一、模（Mod）类（Mod Class）**

你知道，取模运算符（mod operator）可用于求两个数相除后的余数。  
例如：

9 % 5 = 4  
因为 9 ÷ 5 的余数是 4。

我们经常使用 mod 来**将数值限制在某个范围内**。  
例如，如果你的 Spotify 播放列表有 180 首歌（编号为 0~179），  
你当前播放第 175 首歌，想要**跳过 6 首**，那么应该播放第 1 首。  
因为：

175 + 6 = 181  
但没有编号为 181 的歌曲，  
而 181 % 180 = 1。

**二、任务要求**

编写一个类 **ModInt**，用于表示从 0 到某个固定上限（mod 值减 1）之间的数字。  
类中应包含：

* 一个变量表示 **mod 值**；
* 一个变量表示 **当前数值（value）**。

例如：  
如果你想表示“正在播放播放列表中第 175 首歌（共有 180 首）”，  
则可以使用一个对象，其 value=175，mod=180。

你需要实现以下内容：

**（1）构造函数（Constructors）**

* 一个**无参构造函数**；
* 一个**带两个参数的构造函数**。

**（2）访问器（Getters）**

* 提供 getNum() 和 getMod() 等 getter。
* **不要提供 setter**。  
  若想修改 ModInt 的值，请新建一个对象或使用赋值符 =。

**（3）函数定义**

除 getter 外，**所有函数定义必须写在类定义之外**。  
请在 main() 中测试所有函数，并将代码复制到实验报告中。  
在继续下一部分之前，请让助教检查并批准你的代码。

**三、非成员函数 equals()**

添加一个非成员函数：

bool equals(ModInt lhs, ModInt rhs);

当且仅当 lhs 和 rhs 的 **value 与 mod 都相同** 时，返回 true，否则返回 false。  
在 main() 中测试此函数，并将代码复制进实验报告。  
获得批准后再继续。

**四、成员函数 plusEq()**

实现以下成员函数：

void plusEq(ModInt rhs);

该函数会修改调用它的对象，使其变为**自身与 rhs 的和**。  
加法方式为：

(this->value + rhs.value) % mod

**示例：**  
假设你正在播放第 175 首歌，播放列表共有 180 首；  
你想跳过 6 首，则可创建两个对象：

ModInt currentSong(175, 180);

ModInt skipSongs(6, 180);

currentSong.plusEq(skipSongs); // → 结果为 (175+6)%180 = 1

结果是 currentSong 的值变为 1。

**注意：**  
仅能对具有相同 mod 的对象执行加法。  
如果两个对象的 mod 不同，请将调用对象的 value 与 mod 都设为 -1。

💡 **思考题：**  
为什么 plusEq() 必须是成员函数？  
（请在实验报告中回答。）

定义该函数时要写在类外部，并在 main() 中测试。  
获得批准后继续下一步。

**五、非成员函数 plus()**

实现：

ModInt plus(ModInt lhs, ModInt rhs);

该函数返回一个新的 ModInt，表示两者的和。  
提示：实现时可以调用你已经写好的函数（是哪一个？自己思考）。

在 main() 中测试函数，并将代码复制到实验报告中。

**⚠️ —— 至此为最低完成要求（MINIMUM STOPPING POINT）——**

**六、非成员函数 mult()**

实现：

ModInt mult(ModInt lhs, ModInt rhs);

该函数返回一个新的 ModInt，表示两者的积。  
计算方法为：

(lhs.value \* rhs.value) % mod

同样，只有在 mod 相同的情况下才允许相乘。  
若不同，则返回一个 value 和 mod 均为 -1 的 ModInt。

在 main() 中测试该函数，并将代码复制到报告中。  
获得批准后继续。

**七、非成员函数 lessThan 与 greaterThan**

实现：

bool lessThan(ModInt lhs, ModInt rhs);

bool greaterThan(ModInt lhs, ModInt rhs);

* 当 lhs < rhs 时返回 true，否则返回 false。
* 若两者的 mod 不相同，则返回 false。

同样，在 main() 中测试所有函数并将代码复制到实验报告中。