COM2108:功能性编程 - 2022

功能性编程设计案例分析

本文件列出了评分作业的案例研究和你需要完成的任务。它解释了英格玛机（纳粹德国在第二次世界大战中广泛使用的保护敏感通信的机器）和Bombe（由Bletchly公园的破译者开发的机器，帮助破译用英格玛机加密的信息）。具体而言，"炸弹 "的设计是为了发现德国各军事网络中英尼码机的一些日常设置：使用的转子组及其在机器中的位置，信息的转子启动位置，以及插件板的线路。

评分作业的目的是确定你在函数式编程方面的技能发展*得*如何。请记住，通过门槛测试，你已经在该模块中取得了40%的成绩；你在这里获得的任何分数都是额外的。评分任务**是**为了提高难度：模块的平均分数预计约为60%。如果你在评分作业中取得$\frac{1}{3}$，你将在该模块中获得60%的分数--换句话说，普通学生应该期望在这项作业中获得大约$\frac{1}{3}$。仔细检查评分标准，注意你的解决方案的*正确性*只占分数的一小部分。

请注意，本文件中有大量的背景阅读。重点是，你正在接受测试，看你是否有能力解决一个问题并运用功能方法来解决它。你被测试的不仅仅是你编写解决方案的能力，而是你分析问题、开发解决方案和测试你的解决方案是否满足问题的能力。

1 英格玛机器

英格玛机是一种基于转子的加密/解密机。一个转子安装了一个固定的字母替换密码。第一批五个英格玛转轮RI、RII、RIII、RIV和RV的密码为

朴素的	abcdefghijklmnopqrstuvwxyz
RI	ekmflgdqvzntowyhxuspaibrcj
RII	ajdksiruxblhwtmcqgznpyfvoe
RIII	bdfhjlcprtxvznyeiwgakmusqo
RIV	esovpzjayquirhxlnftgdcmwb
巡回演出	vzbrgityupsdnhlxawmjqofeck

图1：一台英格玛机 "达芬奇 "科学与技术博物馆, CC BY-SA 4.0 https://creativecommons.org/licenses/by-tsah/e4.

这些转子的内部接线与这些映射有关，因此在转子I上的A（0）位置的输入将总是给出E的输出，而在转子II上的Z（25）位置的输入将总是给出E的输出。

与Cae-sar密码器（我们在编程练习中看到的）中的移位有略微不同的效果。转子的接线是固定的，所以改变Off集会使转子相对于Cae-sar cypher移位。

其输入/输出位置。例如，图中

图中显示了转子I在其 "原点 "位置（左）和位置1（逆时针1步）（右），以及两种情况下输入 "A "的编码。换句话说，移位改变了转子的输入字母，这迫使信号在转子*内部*通过不同的路径，但同时，*输出信号*也被移位（再次，看图中的右侧部分，其中

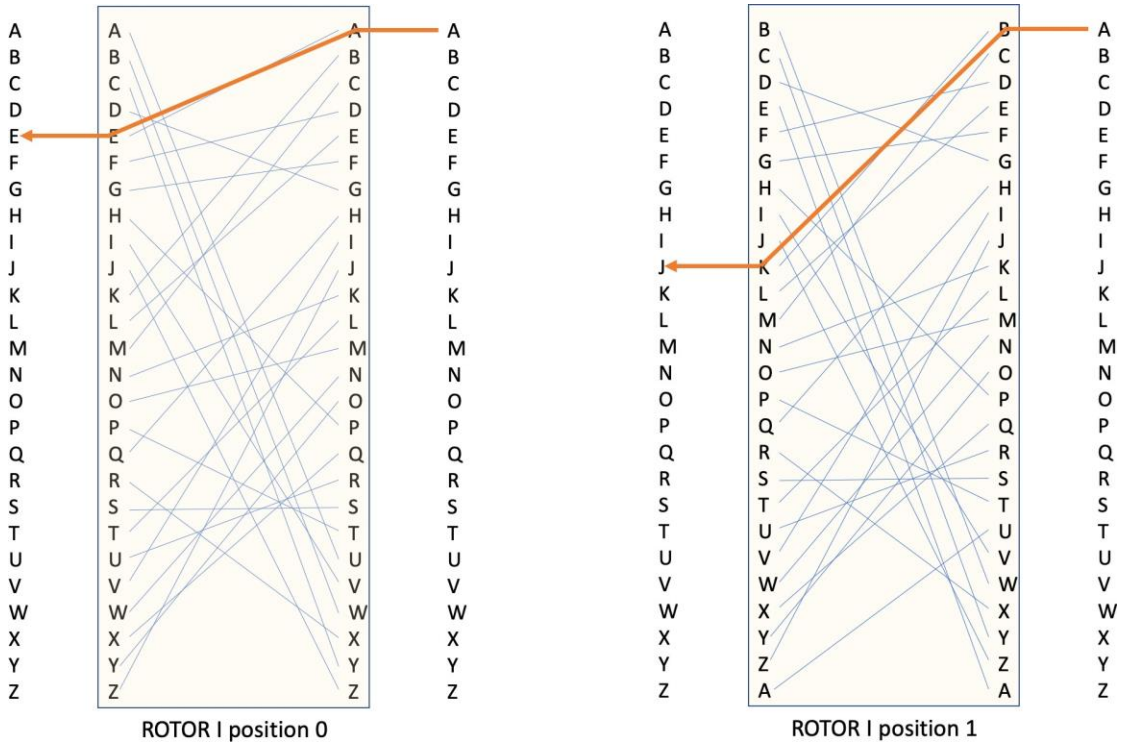

图2：转子I在其 "原点 "位置的接线（左），以及在Off集1（右）的接线。红色箭头显示了每种情况下输入 "A "的输出。

标准的英格玛有三个转子，安装在一个轴上。我们将它们称为左、中、右转子，*LR*、*MR*和*RR*。

给所有英格玛操作员的密码手册规定，每天从可用的转子RI,RII...RV中选择不同的*LR*、*MR*和*RR*。这些都是按照正确的顺序安装在轴上的。此外，还为*LR*、*MR*和RR设定了*OL*、*OM*和*OR*的初始值。

RR。因此，所有使用中的谜题每天都被设置为相同的起始位置。我们假设，在发送每条信息之前，offsets被重置为*OL*、*OM*和*OR*（实际上情况更复杂）。

在基本的英格玛（或称 "无颈 "英格玛--见后面对无颈的解释）中，一个字母首先被传送到*RR*，*RR*对其进行编码并将结果传送到*MR*，*MR*对其进行编码并传送到*LR*。来自*LR*的编码字母被传递给一个固定的**转发器**，转发器进行字母交换（即26个字母被分为13对（c1,c2），其中c1被改为c2，c2被改为c1）。标准的变位器配对（被称为变位器B）是

(A Y)(B R)(C U)(D H)(E Q)(F S)(G L)(I P)(J X)(K N)(M O)(T Z)(V W)

然后，从反射器输出的信号依次通过*LR*、*MR*和*RR*，通过反向编码，传递到输出灯，在这里，编码原始字符的灯被点亮。

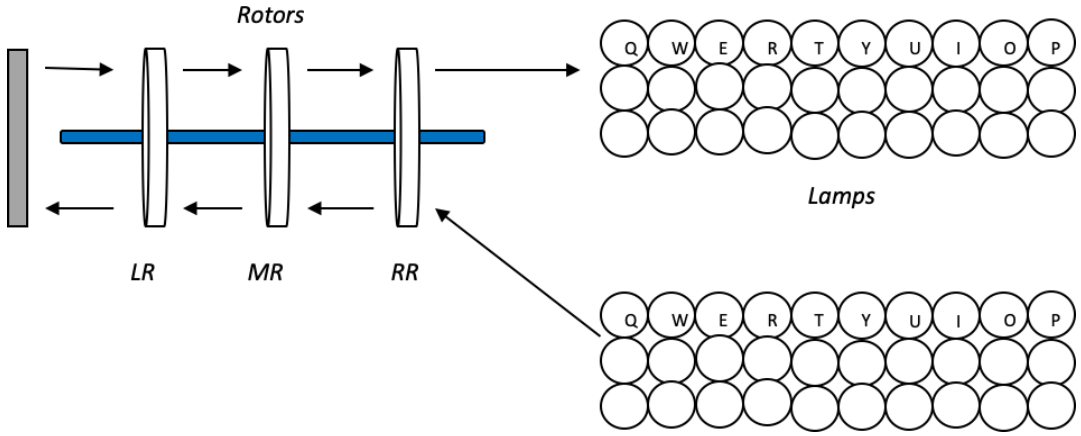

图3：简化的英格玛机示意图

例如，如果*RR*是RI，*MR*是RII，*LR*是RIII，并且offsets都是oi[1]，那么如果A被输入，N将被点亮。

按键	*RR*	*MR*	*LR*	探测器	*LR*	*MR*	*RR*	灯具
A	E	S	G	L	F	W	N	N

请注意，这个过程是**对称的、可逆的**，也就是说，我们刚刚把A编码成N，如果在相同的起点上，我们按N，就会得到A。

按键	*RR*	*MR*	*LR*	探测器	*LR*	*MR*	*RR*	灯具
N	W	F	L	G	S	E	A	A

[1]实际上，当按下一个键时，首先发生的是右转子前进，所以要真正得到这个输出，初始offs必须是（0,0,25）。这将在下一节进一步解释。

1.1 推进转子

每个转子都有一个销子，在每个转子的不同位置，当销子通过时，会使相邻的左边的转子前进一个位置。每按一次键，*最右边的*转子就前进一次。

每个转子的敲击位置如下。

转子	敲打的位置	(数量上)
I	R	17
二	F	5
三	W	22
四	K	10
V	A	0

Bletchly公园的密码分析员使用记忆法中的*皇家旗帜波浪国王以上的*位置重新成员这些位置。

当按下一个键时，在对字符进行编码之前，第一个动作是将*RR*前进1位，即*OR*→*OR*+1，除非*OR*在按下键之前处于**R**位置，在这种情况下

1. $OR \rightarrow (OR+1) \bmod 26$
2. $OM \rightarrow (OM+1) \bmod 26$

因此，当*RR*超过其敲击位置时，*OM*被推进。同样地，如果*OM*的进展超过了它的敲击位置，*OL*就被推进了。如果任何一个转子在Z的位置，下一次前进将把它带到A。

请注意，*在*信号通过转子生成加密字母*之前，*转子的运动就已经发生了。所以在上面的例子中，转子最初处于0,0,25（或A,A,Z）的位置：如果在转子处于这个位置时按下A键，它们会前进到0,0,0的位置，N灯会亮。

1.2 解码

可逆的特性意味着，如果信息的接收者（另一个英格玛操作者）将她/他的英格玛设置为相同的起点，他/她可以通过输入信息来解码 - 原始文本的灯将出现。

1.3 ÄÄÄ

在战时使用中，英格玛被增加了一个**插件板**，其中一对字母被插在一起。如果X被 "钉 "在Y上，那么插板就会输出Y，如果给定了
X，反之亦然。与转发器不同的是，这些配对并不完整：最多有10个配对，其余的字母被传送，没有变化。梗概"，即字母对，每天都会改变。下面是一个例子。

在键盘和英格玛之间以及英格玛和灯之间放置了一块插板。

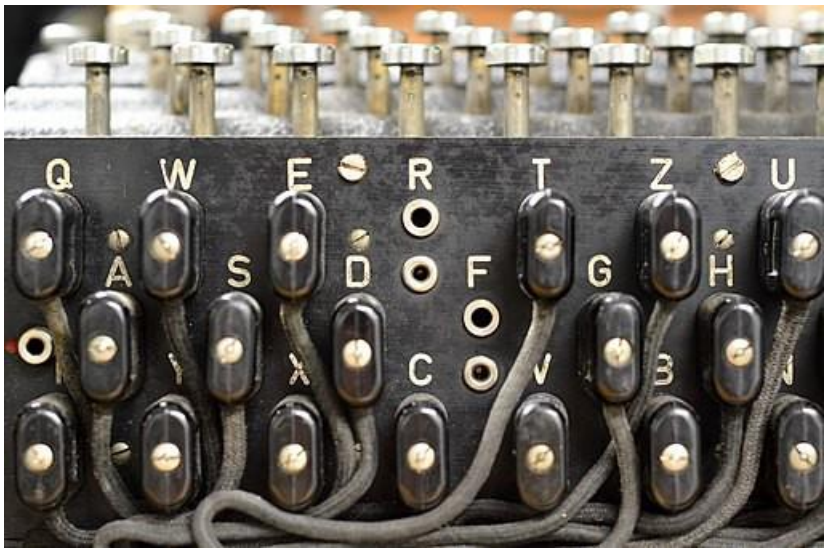

图4：英格玛机的 "Steckerboard"（插板） 曼彻斯特大学数学学院，CC BY 2.0 https://creativecommons.org/licenses/by/2.0，通过维基共享资源。

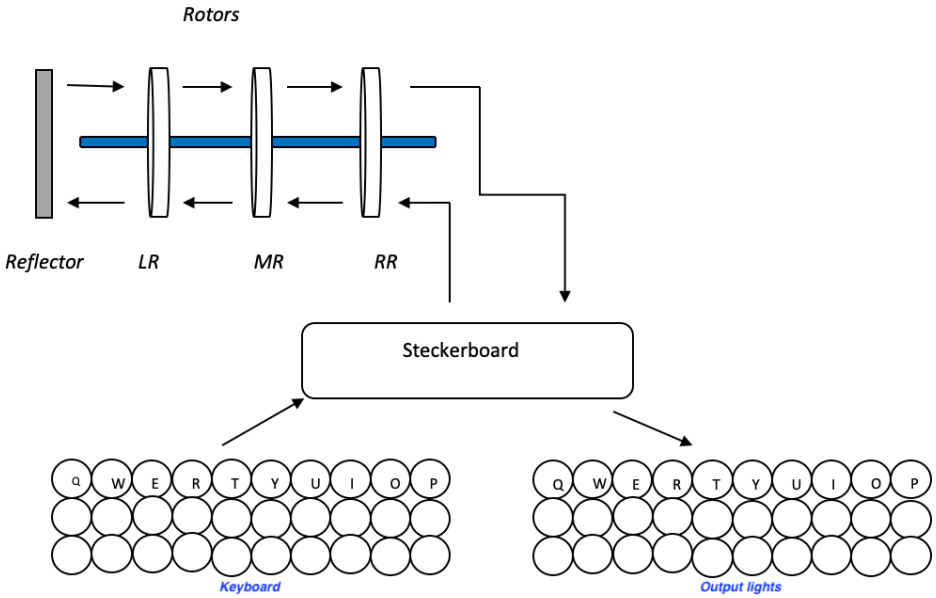

图5：斯特克尔式英格玛机的示意图

一台有条纹的英格玛机仍然是对称的（也就是说，你可以用一台相同的有条纹的英格玛机来破译有条纹的英格玛机的信息）。

2 破解谜团

2.1 破译密码者面临的问题。

布莱切利公园(BP)的破译者知道英格玛机器是如何工作的，他们知道军用版本的机器是有缺陷的。他们还知道5个转子，以及它们的子结构代码，以及反射器对[2].

破译员每天的任务是根据当天传输的信息，从5个转子中选择3个转子，即初始Off集和stecker对。

可能的解决方案的数量约为10^{23} 。当时计算机还没有被发明.....

2.2 炸弹

破译密码的人没有计算机，但他们可以利用从电话交换机上拆下来的部件建造硬件来模拟简单的Enig-mas。这些设备被称为Bombes（以一种冰激凌命名）。图中可以看到一个复原的Bombe。给予转子和初始设定的选择，Bombe可以找到编码的任何字符，并在中间的任何位置上进行编码。

每组垂直的3个表盘对应于英格玛的3个转子。
因此，一个炸弹可以运行
36次模拟。通过26个[3] offsets，花了大约20分钟。

共建造了200多座炸弹，但绝大多数都在战争结束时被摧毁，剩下的几座也在不久之后被摧毁。

图6：Bombe
Ian Petticrew, CC BY-SA 2.0 https://creativecommons.org/licenses/by-sa/2.0, via Wikimedia Commons

2.3 婴儿床和菜单

破解密码依赖于有一个摇篮和一个菜单来指导通过摇篮进行搜索。

阿兰-图灵破解英格玛的方法
编码是基于 **"克里普斯 "的**。军事信息是高度公式化的，可以很好地猜测它们所包含的短语以及信息中的哪些点。
这些短语可能会被发现，例如发件人会在信息的早期被识别出来，

WETTERVORHERSAGE（天气预报）和一个地名可能会在最后出现。一个非常简短的信息很可能是 "没有什么可报告的"。布莱切利公园的专家们能够很好地猜测出克里普斯。

[2]事实上，有不同版本的英格玛机，有些有多达八个转子可供选择。大多数人在任何时候都只使用三个转子，但有些有四个。还有不同的反射器可用。我们考虑到陆军/空军的英格玛机只有5个转子和固定的探测器B，从而稍微简化了这个问题。

这是一个真实的婴儿床的例子。

姿势	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
朴素的	W	E	T	T	E	R	V	O	R	H	E	R	S	A	G	E	B	I	S	K	A	Y	A
密码	R	W	I	V	T	Y	R	E	S	X	B	F	O	G	K	U	H	Q	B	A	I	S	E

破解密码的过程取决于找到一条长长的字母链，在摇篮中连接明文和密码。这条链中索引*i*和索引*j*之间的链接意味着*i*处的密码字符与*j*处的明文字符相同。

例如，在上面，从位置1开始的链是。[1, 0, 5, 21, 12, 7, 4, ...].这样一条链被称为**菜单**。在布莱切利公园，菜单是用手找到的，但你要写一个函数来找出摇篮中最长的菜单。在上面的例子中，[13,14,19,22,4,3,6,5,21,12,7,1,0,8,18,16,9]是长度为17的几个菜单之一。

在Bombe中，暗示追逐过程是用硬件实现的。关 键 部件被称为**对角板**。

2.4 破解密码

假设你假设

- 英格玛转子和反射器的特殊安排（如RI、RII、RIII、反射器B）。
- 一组特定的初始off集，例如（0,0,25）。

你应该找出，对于这些选择，我们是否可以通过菜单为插板找到一套与婴儿床兼容的插头。

- 如果你不能，就改变初始off集的选择。
- 如果你没有初始offs的选择，可以改变Enigma的配置（但你应该使用不要求你这样做的测试案例--你不应该需要编写这个步骤）。

如下图所示，**摇篮对啄木鸟有影响**（与作业2的规格说明中的例子相同）。

姿势	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
朴素的	W	E	T	T	E	R	V	O	R	H	E	R	S	A	G	E	B	I	S	K	A	Y	A
输入	J	E	R	R		T						T						Q					
出	T	J	Q	S		W												B					
密码	R	W	I	V	T	Y	R	E	S	X	B	F	O	G	K	U	H	Q	B	A	I	S	E

假设菜单是[1 ,0,5,21,12,7,4,3,6,8,18,16,9] 。

- 从假设查尔在菜单起始位置的施特克板开始。让我们假设它是无斑点的，即初始Steckerboard是[('E','E')] 。
- 假设谜题在菜单起始位置1处将E编码为J，并给出了off-套。那么（'J','W'）必须被添

加到Steckerboard。

- 按照菜单，这意味着在位置o的时候，英格玛的输入是J，如果它的输出是T，那么就把（'T'，'R'）加到stecker中。

- 在下一个菜单位置，5，假设英格玛将T编码为W，我们尝试将('W','Y')添加到Steckerboard，但你不能这样做，因为我们已经有('J','W')。
- 所以最初的steckerboard[('E','E')]没有成功，我们尝试下一个，例如[('E','F')]。
- 如果你到达菜单的末尾而没有发现矛盾，你就有了一个潜在的解决方案（即初始off集和一个与菜单兼容的steckerboard）。
- 如果所有26个初始Steckerboard对都被试过了，我们就改变初始off集，再试一次。

3 你的任务

你需要为Enigma机器和Bombe编写模拟器（以简化的形式）。**使用Enigma.hs中提供的存根。**仔细研究Main.hs中函数的调用方式：该文件显示了参数传递给你所要编写的函数的方式。(你还需要写更多的函数，但encodeMessage、longestMenu和breakEnigma这三个函数将被测试）。如果你的类型与自动测试中使用的类型不一致，你的代码将不会被编译。

如果你提交的代码不能编译，你的作品中的实施部分将得到0分。

3.1 英格玛的模拟

在顶层，你必须有一个函数encodeMessage。在存根文件中提供了类型的定义，同时还提供了代表Enigma机器规格的类型定义。

```
数据 Enigma = SimpleEnigma Rotor Rotor Rotor Reflector Offsets
            | SteckeredEnigma 旋转器 旋转器 反射器 偏移量 Stecker
```

子组件（转子、驱动器、Offsets、Stecker）的类型定义*没有*正确提供。你需要根据Main.hs中的示例用法来提供这些。

看看encodeMessage的预期用途（如Main.hs所示），然后编写适当的 类型定义。但是，先不要开始写任何其他的代码。

3.1.1 记录设计

正如在讲座中所讨论的，**在你考虑代码之前**，你应该先考虑设计。对于这项作业，你必须创建一份报告，在报告中清楚地解释你的解决方案的设计（图表可能有帮助）。

3.1.2 实施你的设计

当你完成了你的设计，你应该实现它。一个好的设计应该比较容易转化为代码；如果你觉得实现起来很困难，你应该考虑回到你的设计中去，并在其中加入更多的细节。然而，不

要

试图一下子就实现它!你应该仔细考虑你实现函数的顺序以及你如何测试它们。(见下一小节)。

当你实现你的代码时，记得使用良好的功能风格，选择合理的名称（为函数、参数和类型），并*在需要的地方*加入注释。请注意"*需要的地方*"--写得好的代码应该是不言自明的。简短的注释说明大多数函数的目的是一个好主意（但你可能有一些非常简单的低级函数，不需要解释），头文件（在文件的顶部）是必不可少的，但你一般不需要对每一行代码进行注释（只有在不清楚它们是做什么的时候）。我们鼓励你在实验课上询问有关风格和注释的反馈。

3.1.3 记录你的测试

在你创建的文件中添加一个章节，以记录你的设计。在这一节中，明确指出。

1. 你实现你的功能的顺序。
2. 对于每个函数。
    - 对该功能进行测试的基本原理
    - 一些说明性的例子，显示测试的输入和输出。这组（小的）例子应该清楚地表明你的函数是按计划工作的。(在这里用一个表格就可以了！)

仔细考虑你的测试。我不希望看到你所做的**每一个**测试，但我希望看到你选择了一个适当的测试数据范围的证据，并且你的功能在一般情况和边缘情况下都能工作。

3.2 寻找最长的菜单

如前所述，*婴儿床在*破解谜语密码中发挥了重要作用。(见第**1**节)对于这部分作业，你必须。

1. 为Crib和Menu定义数据结构
2. 写longestMenu :: Crib -> 菜单，它被赋予一个**crib**并返回最长的菜单。

至于前一个阶段，你应该。

1. 记录你的设计
2. 实施你的设计
3. 记录你的测试

3.3 模拟炸弹

为了简化事情，假设转子和反射器的选择始终是相同的：左边的转子将是RI，中间的转子是RII，右边的转子是RIII。转子是

标准的一个（这被称作是反射器B）。一旦你有了工作代码，你可以尝试重新移动这些限制。

请注意，你正在编写一个搜索代码，一个可能持续很长时间的搜索。因此，你应该发明一些能够快速找到解决方案的测试案例--特别是要仔细考虑合理的初始转子位置。如果我们已经建立起了模拟有桩谜题的函数，我们就可以很容易地构建这样的测试案例。

你的挑战是写一个函数breakEnigma :: Crib -> Maybe (Offsets, Stecker)将在给定的摇篮中搜索一个解决方案。请注意，该函数有一个*也许的*返回种类，因为它可能无法找到一个解决方案!

至于前几个阶段，你应该。

1. 记录你的设计
2. 实施你的设计
3. 记录你的测试

请注意，设计是**绝对**关键的。(我的意思是，它总是如此，但如果你不仔细考虑设计，就不可能完成这个阶段！)如果你遇到了困难，试着从代码中完全切换出来，把注意力放在算法上。

3.4 批判性思考

现在你的报告中应该有三个主要部分。

1. 模拟英格玛
2. 寻找最长的菜单
3. 模拟炸弹

每个部分都将描述该组件的设计和测试（或至少是你设法实现的该组件的部分）。

报告的最后部分应该是对你的经历的**批判性反思**。什么是 "批判性反思"？它是基于这样的理念：我们所有的经历都应该是一个持续的反思循环的一部分。

在这最后一节中，你应该写一篇**简短的**（不超过半页）批判性的反思，挑出一两个使用函数式编程的具体经验，并讨论你将如何使用这些经验以及你对这些经验的反思，在你未来更广泛的活动中发挥作用。

4 提交

你必须为这项作业分别提交两份材料。

1. 报告，以Microsoft Word或基于文本的PDF文件形式提交，使用Blackboard上的适当链接（报告）（该文件的名称并不重要），以及

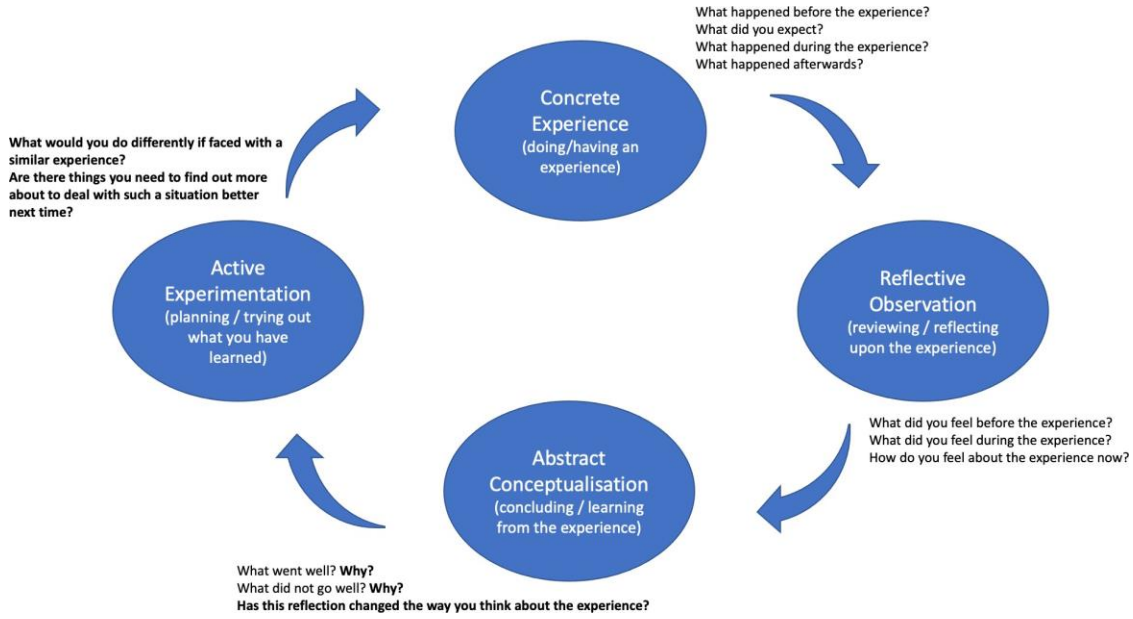

图7：Kolb的体验式学习周期

改编自：Kolb, David A. 经验式学习。经验是学习和发展的源泉。FT出版社，2014年。

2. 一个单一的源文件，Enigma.hs，在一个叫Enigma的模块中实现你的解决方案（根据存根）。这个源文件将被自动测试，就像黑板上提供的Main.hs例子一样（但自动测试的输入将与该例子中的内容不同）。同样，请确保你使用Blackboard上正确的（实施）链接。

5 评估

你的代码和你提交的报告都将被评估。对于你的代码。

- 它应该产生正确的结果。 encodeMessage应该正确编码/解码消息。
- 它应该表现得很好。这包括功能的组织、行的布局、名称的选择、为清晰起见适当使用的空白。
- 它应该有适当的注释--标题注释、功能注释，以及适当的内联注释。

为你的报告。

- 它应分为所述的四个分节。
- 对于前三节中的每一节。

  - 设计必须是清晰的，包括功能和数据流。
  - 测试的描述应该指出你实现这些功能的*顺序*，并使我相信你已经认真考虑了如何适当地测试每个功能（并且已经完成了测试！）。

- 在最后一节，你将根据你如何清楚地表达你的批判性再思考和思考的相关性来评分。

元素	检查了	分数的比例
报告		
设计	对于每一个部分（编码信息、最长的菜单、破解密码），从高层次的问题到低层次的问题，都有明确的分解。 职能。	30
测试	对于每一节，要明确说明函数的执行顺序（显示出合理的顺序选择），并证明你已经考虑了一般和特殊的情况 你的测试集中的案例。	20
批判性思考	你已经确定了一个特殊的经验，并解释了这是如何改变你的长期实践的。我特别希望看到的是，有证据表明这一点可以转移到这个特定模块之外的实践中。	10
实施		
正确性	编码信息	5
	最长的菜单	5
	破解英格玛	10
风格	合理的命名规则、布局、功能排序、行的长度、一般的可读性------。 它	10*
评论	标题和（合理的）函数注释包括在内，必要时使用内联注释（太多和太少一样糟糕）。	10*

表1：评分作业的分数分类

注意事项。

1. 如果你提交的代码不能编译，你的执行部分将得到0分。

2. 风格和注释的分数将与你的代码的*正确性*成比例。如果你的encodeMessage得了5/5分

，最长的Menu得了1/5分，break- Enigma得了2/10分，风格得了8/10分，评论得了5/10分，那么你在实现部分的分数将被计算为。

$$(5 + 1 + 2) + ((8 + 5) \times \frac{(5 + 1 + 2}{20})) = 8 + 5.2 = 13.2$$

(而**不是**从结果的直接相加中产生的**21**个）。