2017~2018 秋季学期编译原理实验

Decaf Compiler 实验总述

什么是 Decaf？

Decaf 是一种非常简单的面向对象编程语言。它是一种强类型的、面向对象的、支持单继承和对象封装的语言。实验用的 Decaf 更加类似 Java，与 C++有比较大的差别。学会用 Decaf 写程序是非常简单的一件事情，但是请记住 Decaf 跟现实中使用的编程语言并不完全相同，它是经过简化且面向编译器教学的需要构造的。下面是一段 Decaf 程序：

```
class Main {
    static void main() {
        class Fibonacci f;
        f = new Fibonacci();
        Print(f.get(ReadInteger()));
    }
}

class Fibonacci {
    int get(int i) {
        if (i < 2) {
            return 1;
        }
        return get(i - 1) + get(i - 2);
    }
}
```

这段代码的大意是从键盘读取一个整数，然后把下标为这个整数的 Fibonacci 数打印到屏幕上。从中可以看出：

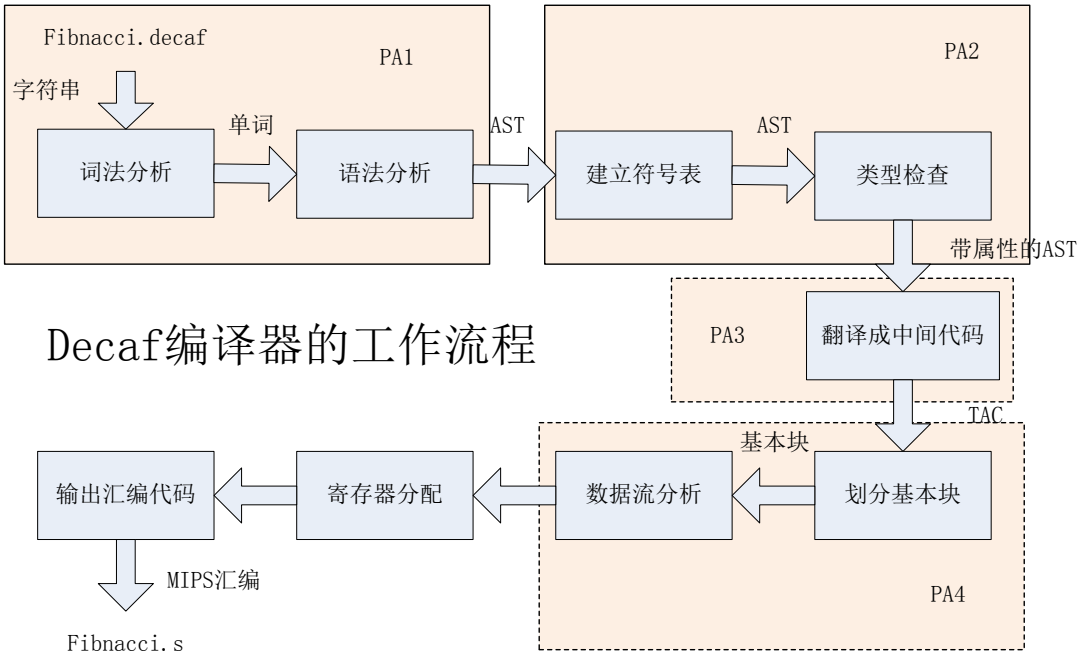
1、Decaf 程序有一个叫做 Main 的类，并且含有一个静态的，返回值为 void，参数列表为空的 main 函数，这是整个程序的入口。main 函数必须是静态函数，返回类型为 void，参数列表为空。

2、Decaf 程序中引用类名、函数名等等不需要有事先声明，但是所引用到的符号在整个程序中必须有适当的定义（这一点跟 Java 是一样的）。

Decaf Compiler 实验要做什么？

亲自动手，实现一个非常简单的 Decaf 编译器。这个编译器的输入是 Decaf 源语言程序，输出是 MIPS R2000/R3000 伪指令汇编程序。

下图是 Decaf 编译器的大致工作流程：



Decaf编译器的工作流程

我们把这个编译器的实现划分为 5 个阶段：

1、 词法分析、语法分析及抽象语法树生成（PA1）

本阶段可分为两个子任务，一是词法分析，二是语法分析。抽象语法树（以下简称语法树）的生成采取语法制导的方法，因此将其划归到语法分析子任务中。

词法分析的功能是从左到右扫描 Decaf 源程序，从而识别出标识符、保留字、整数常量、算符、分界符等单词符号（即终结符），把识别结果返回到语法分析器，以供语法分析器使用。在识别的过程中，我们还需要检测词法相关的错误，例如字符@并非 Decaf 程序中的合法符号，若这个字符在注释以外出现，则需要向用户提示一个词法错误。

语法分析是在词法分析的基础上对词法分析得到的终结符串建立语法树，并对不符合语法规则的 Decaf 程序报错。比如常见的少写分号的问题，就属于语法错误，会在这个阶段被发现。

PA1 的最终结果是一棵跟所输入的 Decaf 源程序相对应的语法树。

在我们的实验中，PA1 的实现分两种方案，分别对应两项实验内容 PA1-A 和 PA1-B.

PA1-A 的重点是掌握 LEX 和 YACC 的用法，体会使用编译器自动构造工具的好处，并且结合实践体会正规表达式、自动机、LALR(1)分析等理论是如何在实践中得到运用的。

PA1-B是通过手工方式实现词法分析、语法分析及语法树的生成，不再使用LEX/YACC。本学期中，PA1-B 的具体要求参见对应的实验导引文档。

2、 语义分析（PA2）

能够成功建立语法树只说明了所输入的 Decaf 源程序在格式上是合法的，但是要进行有效的翻译，编译器还需要了解这个程序每个语句的含义。了解程序含义的过程称为语义分析。考虑下面程序片断：

```
int str;
str = "abc";
```

这个程序是符合 Decaf 语法的，可以通过 PA1 的检查并建立语法树，但这段程序显然是不正确的。

在 PA2 中，我们把语义分析过程分为两个内容：分析符号含义和检查语义正确性。分

析符号含义是指对于表达式中所出现的符号，要找出这个符号所代表的内容，这个工作主要通过检索符号表实现。检查语义正确性指的是要检查每个表达式的操作数是否符合要求，也就是说这个表达式是否是语言规范中所规定的合法的表达式。由于不合法的语句具体含义在语言规范中没有规定，从而使得编译器没法明确这些语句的确切含义，所以检查语义的正确性是很有必要的。

如果一个程序成功通过语义分析，则说明这个程序的含义对于编译器来说是明确的，从而翻译工作才能得以进行。

3、翻译成中间代码（PA3）

由于源语言和目标语言一般有比较大的差别，因此直接把语法树翻译为目标语言中的合法程序通常是比较困难的。大多数编译器实现中所采取的做法是首先把源语言的程序翻译成一种相对接近目标语言的中间表示形式，然后再从这种中间表示翻译成目标代码。

在 Decaf 编译器中，我们采用一种叫做三地址码（Three Address Code，即 TAC）的中间表示形式。在 PA3 中我们会对这部分的内容进行更加详细的说明。

阶段三完成后，三地址码程序可在实验框架中给定的 TAC 模拟器上执行。

4、数据流分析与代码优化（PA4）

一般来说，在三地址码的基础上是可以直接翻译为目标代码的，但是这样的直接翻译会导致所产生的代码的效率比较差，所以多数编译器都会进行一定的优化工作。大多数编译优化的基础是数据流分析。所谓数据流分析，是指分析各种数据对象在程序的执行路径中的状态关系，例如一个变量在离开某个语句以后是否还有用等。

在 PA4 中，我们目前的课程实验只要求基于 TAC 实现简单的数据流分析。具体要求参见对应于这一阶段的实验导引文档。

5、目标代码生成（PA5）

PA5 的核心内容是生成汇编代码，实验框架中主要包括汇编指令选择、寄存器分配和栈帧管理等模块。经过以上四步以后得到的中间结果可以通过 PA5 转化为 MIPS 汇编代码。考虑到学生负担问题，目前的编译原理课程实验中没有安排这一阶段的实验任务。

完成这些阶段以后，即可产生出适合实际 MIPS 机器上的汇编代码，可以使用 Wisconsin 大学所开发的 SPIM 模拟器来运行生成的代码。

目前，编译原理课程实验未涉及 PA4 的代码优化和 PA5 的目标代码生成相关的内容。这些内容将会在后续课程《计算机系统综合实验》中有相应的要求。

你们的起点是什么？要做到什么程度？

虽然这个实验只是一个很简单的编译器，但是由于你们的实验时间有限，因此我们已经提供了一个相对比较完整的实验框架，这个实验框架包括以下主要内容：

1、除 PA1、PA2、PA3、PA4 要求以外的所有代码，特别是相关的数据结构

2、各阶段的详细说明文档

3、各阶段的部分测试例子

4、用于建立编程环境的工具

5、……

尽管如此，大家仍然需要根据所处的项目阶段，分别独立完成所要求的内容。

实验环境

实验中会用到下列工具：

(1) Java：我们的编译器使用 Java 实现，请自行安装 JDK（1.6 或 1.7），并建议安装 eclipse。

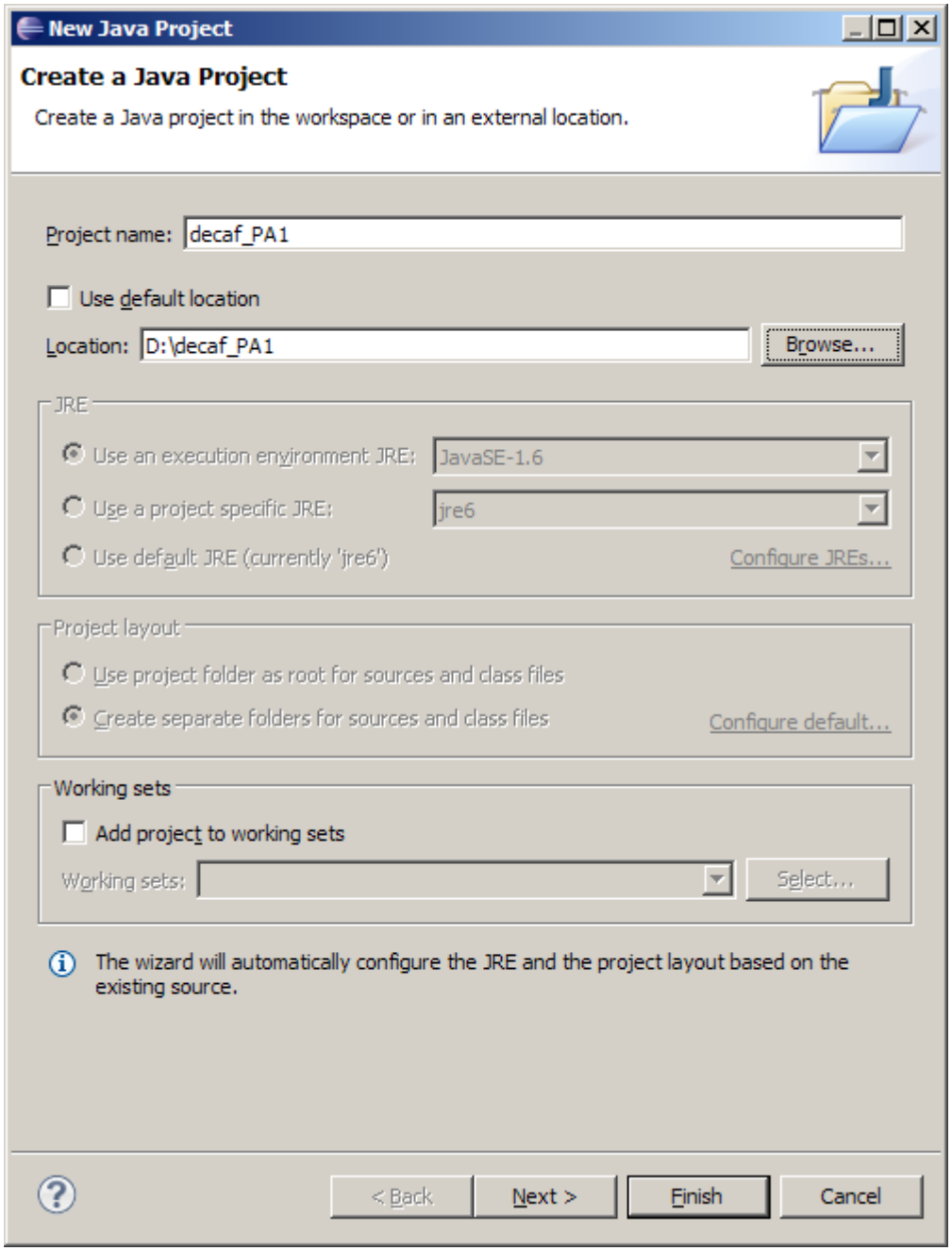
(2) JFlex、BYACC/J 和 SPIM：直接包含在发布的框架中，不需要单独下载。我们提供了 Windows、Linux 和 Mac 三种版本。

(3) Python：请自行安装 Python 2.7 或 3.2。考虑到访问 Python 网站部分页面会遇到 Connection Reset，为节省大家的时间，第一阶段的框架里提供了 Python 3.2 的 Windows 安装文件。

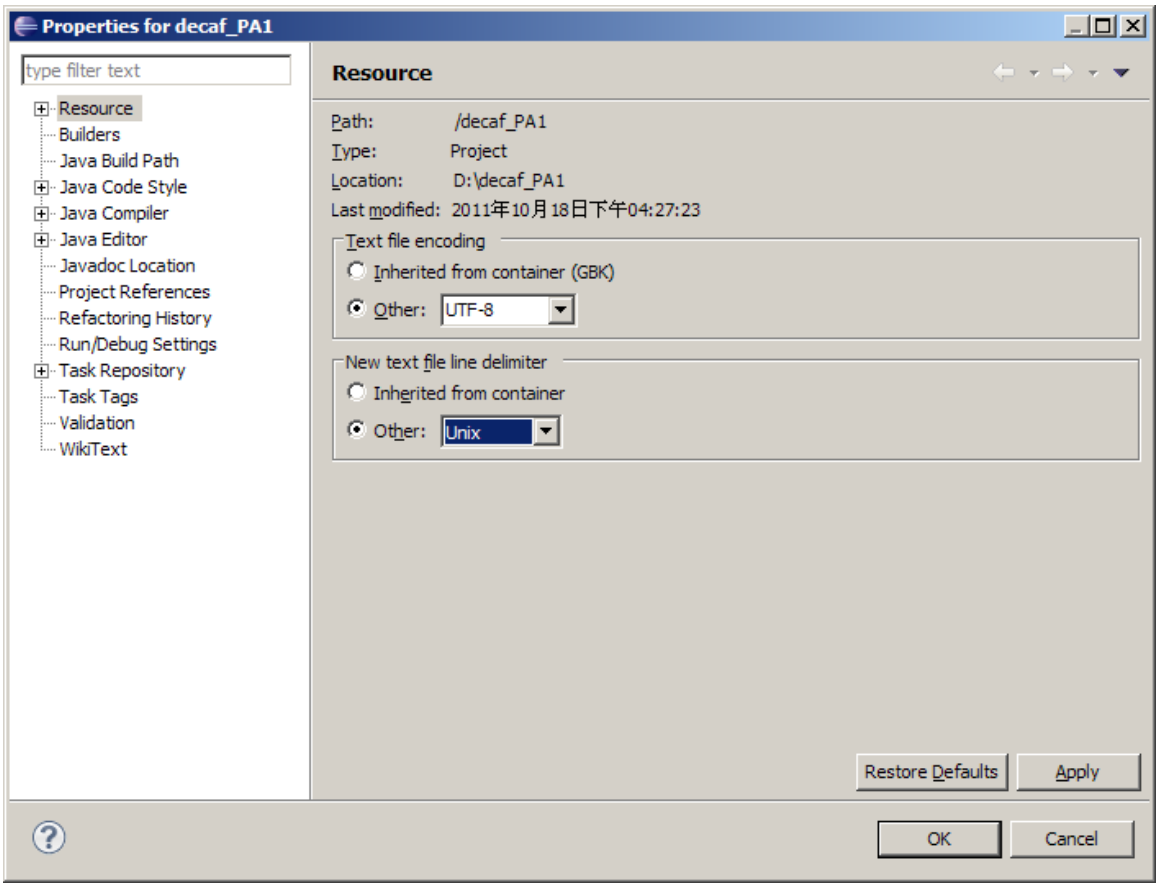
使用 eclipse 搭建实验环境的步骤如下：

1、从网络学堂下载实验框架，解压，例如解压到 D:\decaf_PA1。（路径里最好不要包含中文和空格，有可能遇到问题。）

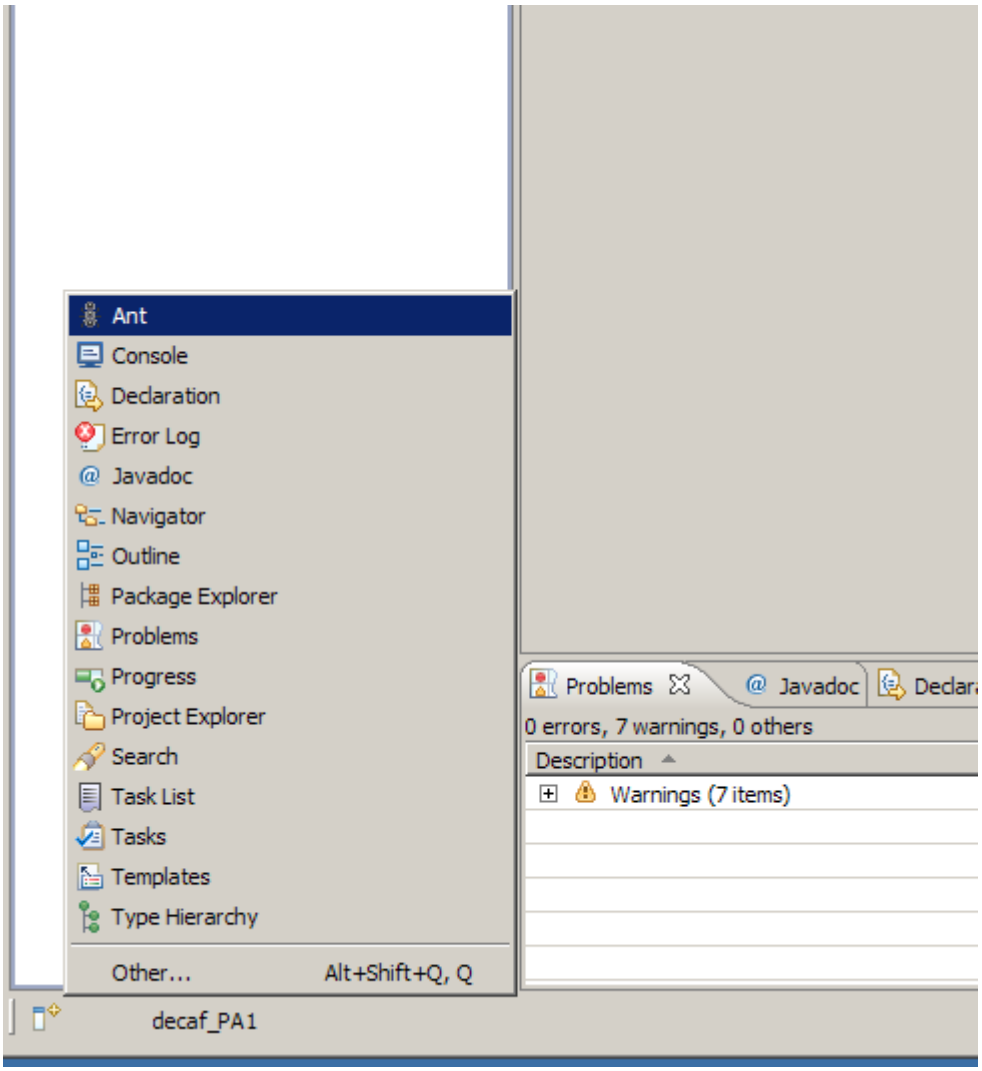
2、启动 eclipse，选择 File → New → Java Project，选择解压后的文件夹作为工程目录，起好工程名之后点 Next、Finish，工程就建好了。如下图：



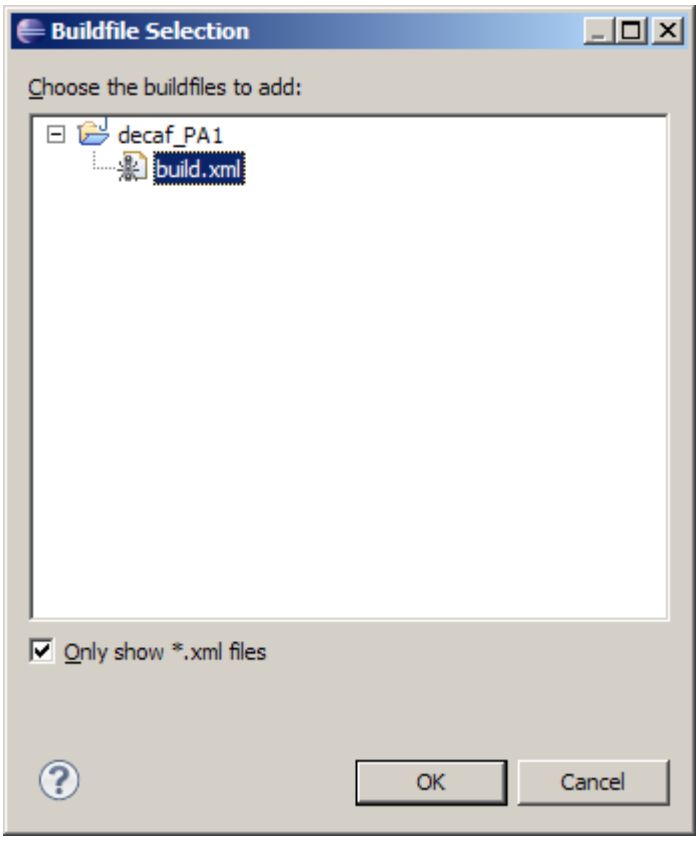
3、（使用 Linux 的同学跳过此步）右键单击工程名，选择 Properties，将文件编码设置为 UTF-8，文本文件换行符设置为 Unix，如下图：



4、接下来开始配置 Ant。单击左下角的 Show view 按钮，在弹出的菜单中选择 ant，会出现 ant 的工作目录，如下图：



5、在 ant 工作目录中，右键选择 Add buildfiles，选择 build.xml 文件，如下图。之后就可以在 ant 的工作目录中操作 decaf 了，双击对应的 task 就可以执行。



测试方法

测试样例放在 TestCases/S*文件夹中。运行其中的 runAll.py 即可测试（测试前记得在 eclipse 中编译）。runAll.py 也接受参数，可以指定只测试其中部分样例，如在命令行运行 runAll.py test1.decaf test2.decaf 将只测试这两个样例。

我们会保留一部分测试例子不公开，因此一定要自己编写一些测试例子来验证你的程序是否正确。

提交方式

完成实验的代码部分以后，还需写一份实验报告，说明你是怎么样完成该作业（例如加入了哪些新的数据结构或者函数、程序的工作逻辑等）以及作业过程中遇到的问题和解决方法等。简要说明即可，不需要太详细（一页纸以内）。需要特别注意的是，如果在自己的程序中借用了别人的成果或者思路，请在报告中明确说明，并指出参考的内容。请将实验报告命名为 report.txt（或 doc、docx、pdf），放在代码根目录（如 decaf_PA1）下。

完成上述步骤后，运行 submit.py，它会将 decaf.jar 和你的实验报告一起打包为 submit.zip 文件，将这个 zip 文件上传到网络学堂。由于课堂人数较多，为了便于测试和评分，请一定按此方法操作，不要自行打包。

以上仅适用于 Decaf 前四个阶段的实验。此外，可能会有同学选作特定的拓展实验，

将在第四阶段之后再安排时间提交和评分。

如果发现实验框架有任何问题，请告知助教，以便及时解决问题或更正。

评分标准

Decaf 每阶段实验所占最终成绩的比例见课程第一讲课件。程序部分主要看输出结果与标准输出的一致程度，占每阶段成绩的 80%。报告部分主要看所提交的作业报告的描述，例如是否清楚说明了自己的工作内容等，占每阶段成绩的 20%。

若有拓展实验，则在四个阶段的评分完成后统一评分。拓展实验的评价是综合考虑创新性、实用性、合理性、难度、工作量等因素进行的。

在每阶段截止提交以后，我们一般会在两周内在网络学堂上公布成绩。如果你认为成绩有问题，请及时与助教联系（关于实验成绩的疑问最晚请在期末考试以前提出，考试以后不能再更改实验成绩）。

由于实验有一定难度，同学之间相互学习和指导是提倡的。对于其他同学的代码，可以参考，但不可以直接拷贝。如有代码交给其他同学参考的，必须向老师或助教申明，告知给哪些同学拷贝过代码（包括可能通过间接渠道传播给其他同学）。如发现有代码拷贝的情形，拷贝者和被拷贝者将会得到同样的处罚，除非被拷贝的同学提交时已做过声明。