

week6 实验：程序流程图和伪代码

15331302 王翔宇 软工2015级1班 教务一班

MarshallW906@GitHub

week6 实验：程序流程图和伪代码

Overview

对流程图和伪代码的理解&看法

1.流程图

流程图制作工具：

常见/常用流程图形状

常用流程图组合

Process On常用操作技巧

2.伪代码

对伪代码的认识和感想

本次实验题题目

Assignment 0

Assignment 1

Assignment 2

Assignment Ex

option 1

option 2

Discussion , Cooperation and Summary

option 1 : Github Discussion

option 2 : Simple Presentation

option 3 : Helping Others

option 4 : Personal Summary

Self Grading

结语

Overview

学习用程序流程图和伪代码来描述算法，方便梳理思路和与他人交流。

对流程图和伪代码的理解&看法

~都是可以比较直观的表达程序设计思路的工具。
~流程图更直观，有助于快速理解代码思路；
~伪代码则可以快速、方便地被重写成真正用于编译的代码。
~在面对一些比较复杂的题目时（诸如USACO前两章这种大型模拟题以及复杂算法题等等），先写一个大致的流程图/伪代码将比直接写的效率高很多。

1.流程图

流程图制作工具：

~<https://www.processon.com/>

免费在线制作流程图的网站。功能强大，UI方便。不能再赞。感谢TA提供干货。

~Diagram Designer

找到的一款PC软件，似乎很古老，而且在win8.1上存在一些bug。

·二者相比，前者完全压制后者（有网络时）。

常见/常用流程图形状

常用的流程图图形大约有如下几种：

开始/结束

条件判断

输入

数据操作/流程

展示/输出(如print)

~图形内均可添加文字。

~~文字内容与伪代码类似。

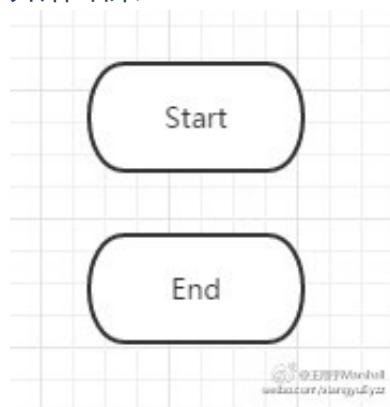
~~以简洁、易读为好。

~此外还有带箭头和不带箭头的连线。

~~连线上可以添加文字。

~~连线遇到拐点时，拐角一律为90度。

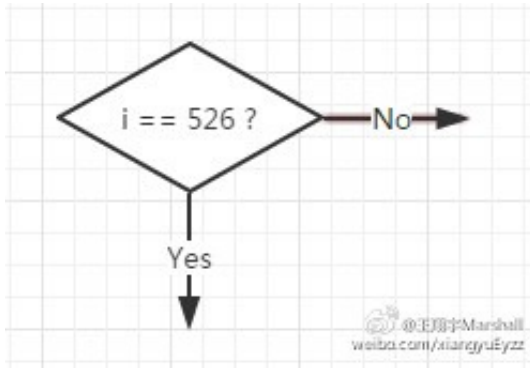
开始/结束：



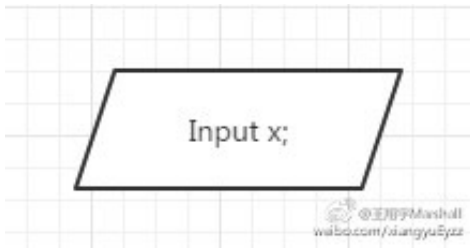
条件判断：

条件判断框内的文字通常是一个计算值为true或false的表达式 + 一个英文问号。

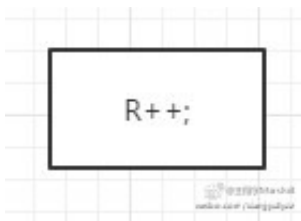
一般情况下，箭头从上方进入，Y、N两分支可从任意方向画出，但仍以易读、清晰为好。



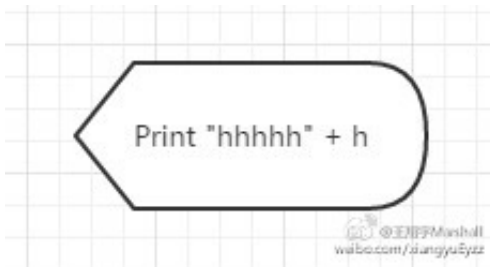
输入数据：



数据操作/流程：



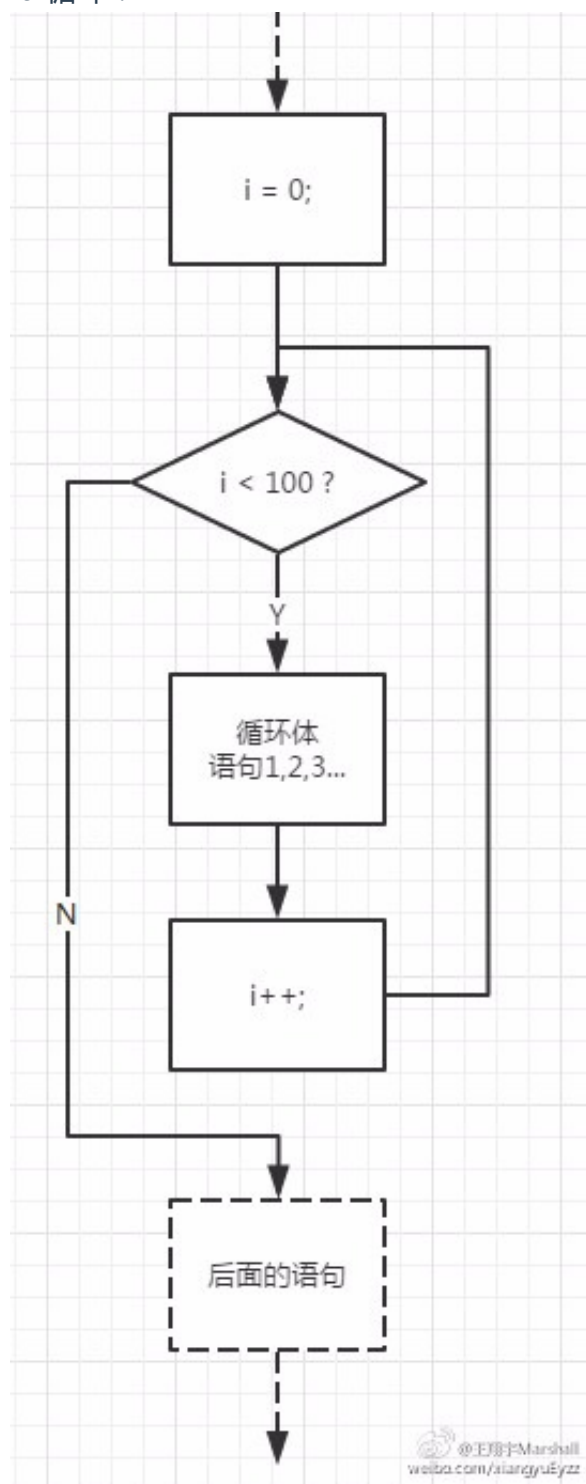
展示（比如到stdout上）：



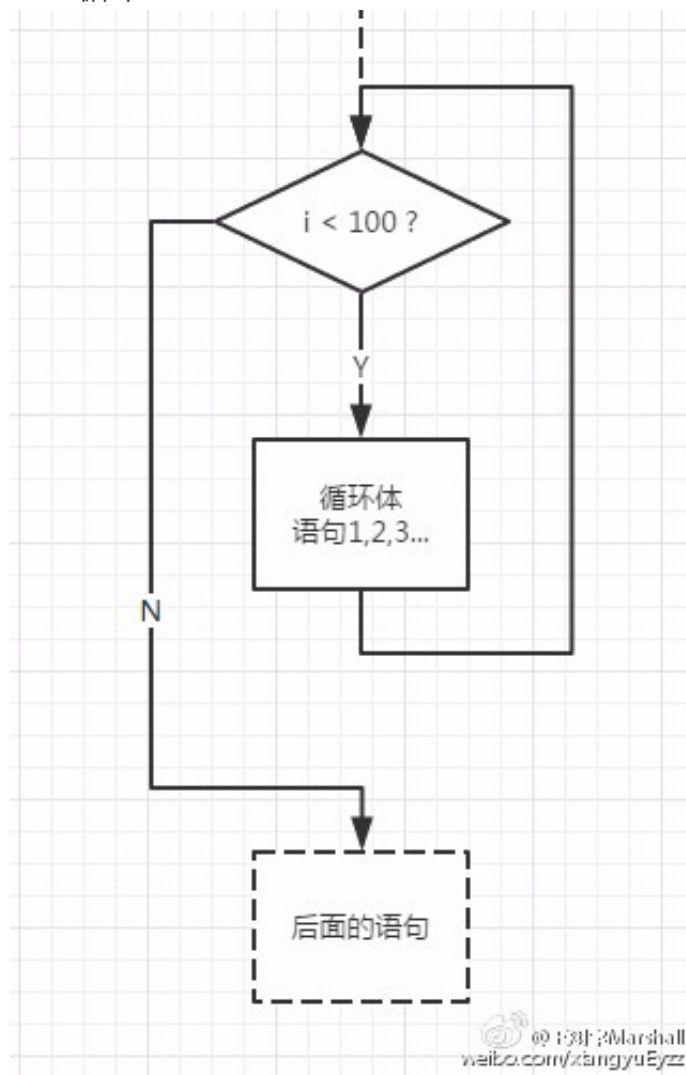
常用流程图组合

以下分别是for循环、while循环、do-while循环的流程图示例：

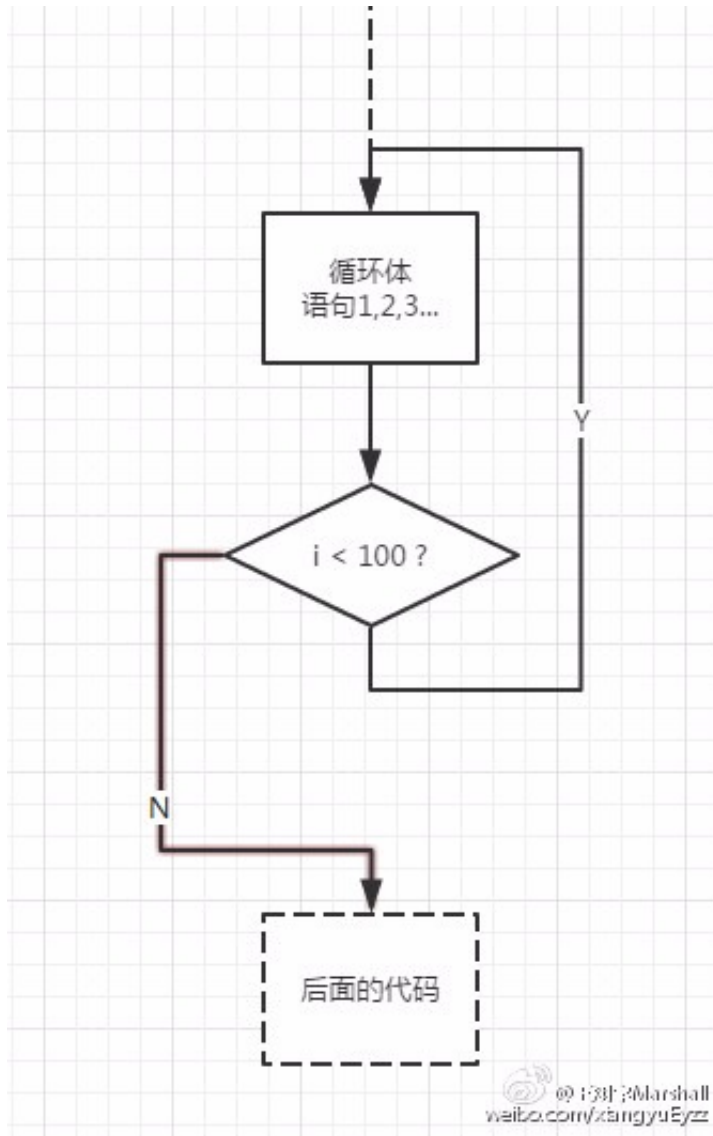
for循环：



while循环：



do-while循环：



Process On常用操作技巧

通用：

Alt 按住Alt，通过鼠标可以对页面进行拖动

Ctrl 按住Ctrl，点击一个图形，将其添加到选择图形中，或者从中移除

Ctrl + (+)，Ctrl + (-) 放大，缩小

Ctrl + A 全部选中

Esc 取消选中，并取消当先操作

T 插入文本

I 插入图片

L 插入连线

图形被选中时

箭头 (←↑↓→) 将选中图形向左、向上、向下、向右移动

Ctrl + 箭头 (←↑↓→) 每次微移一个像素

Ctrl + 调整大小调整图形大小，并且约束比例

Ctrl + Z 撤销

Ctrl + Y 恢复

Ctrl + X 剪切

Ctrl + C 复制

Ctrl + V 粘贴

Ctrl + D 复用

Ctrl + Shift + B 格式刷

Delete, Backspace 删除

Ctrl +] 将选中的图形置于顶层

Ctrl + [将选中的图形置于底层

Ctrl + Shift +] 将选中的图形上移一层

Ctrl + Shift + [将选中的图形下移一层

Ctrl + L 锁定选中的图形

Ctrl + Shift + L 将选中的图形解锁

Ctrl + G 组合选中的图形

Ctrl + Shift + G 将选中的图形取消组合

编辑文本

空格 编辑文本

Ctrl + B 粗体

Ctrl + I 斜体

Ctrl + U 下划线

Ctrl + Enter 保存文本编辑

2. 伪代码

对伪代码的认识和感想

原则上讲，只要可以将意思传达清楚，伪代码可以用任何形式的语言（语法）表达。

有的人由此选择以类似自然语言的方式书写，以减少书写代码时由严苛的规则带来的压力。

但我认为这种方式其实意义不大。对于思考规模比较小的习题，整理思路的过程在大脑中就可以顺利完整地进行；对于规模较大的问题，与其用自然语言描述，不妨先用流程图画一个较为直观的导图。

在这种情况下，我把伪代码当作是一种“虽然要求不严苛，但要有一定语法规则”的思路表示形式。这样可以进一步理清自己解决问题的思路，同时也减少了将其改写成真正代码所用的思考、翻译时间。

*个人认为，Pascal、VB之类十分易学的语言，其语法经过修改、简化至较自由的版本后，十分适合写伪代码。

**大括号{} 单双引号"" 方括号[]等实用标记，最好保留。

本次实验题目

Assignment 0

本题为自选一道本周sicily上的课堂习题，写出其伪代码。
我的选择是1008，求素数的题目。

Assignment 1

本题为给定流程图，写出伪代码和C程序代码。

整题浏览后，我首先想到的是和给定流程图思路不同的算法。在得到TA叶师兄的允许后，我决定用自己的算法做本题。所以在Assignment1文件夹内的资源均为我独自算法的代码和伪代码，以及一份对应的流程图（Flowchart.png）。

对本题的感想和建议


1. opt1和opt2内容好像重复
2. 题面给出的流程图中的算法是错误的。错在计算subordinates一步。根据本题信息，可以把按符合要求逻辑排列过后的所有员工及其关系看作一个树。对于每一个拥有分支节点作为子节点的元素（员工、一部分人的上级）、和排列的最后一条元素（员工）来说，他们的下属数量（subordinates）将全部计算错误。按照给出的算法，求得的subordinates为该节点的度，而正确答案应该为该节点下所有子节点的个数（也即所有子节点度的求和）。
3. 流程图中还有一个比较模糊的地方，就是输出的boss实际是指immediate boss，不过很容易推出是这个结果，不知是否需要增加这个说明。
4. （接第2条）如果按照这个方式求结果，无疑会让问题无端变得麻烦。既然已经按薪水从小到大排序，我的想法是：从第一条遍历至被查id的前一条信息，所有height <= height_current 的数目即为其下属数目（subordinates）；从被查id的后一条信息遍历至末尾，所有height >= height_current 的信息的数目即为其之上的上司数目(虽然本题用不到)，而向后遍历时，查到的第一条符合要求的信息即为其immediate boss。
5. 抛开错误不谈，本题的思路是先把所有用户可能查询的资料全部计算并记录，然后检测之后的查询命令直接输出结果，是一种类似数据库的方式。我的操作顺序是：读入数据→排序→读入查询指令→计算结果→输出。细想来，若仅供单次使用，那么我的方法总效率显然更高；但若是类似的、较大的、需要多次使用的项目，则显然是原算法的思路顺序更有效。

附自写程序的几个测例：


```
F:\rustbin\ProgDesignI\week6shiyan\Assignment1>source.exe
2
3
123456 14323 1700000
123458 41412 1900000
123457 15221 1800000
123458
(0,2)
4
200002 12234 1832001
200003 15002 1745201
200004 18745 1883410
200001 24834 1921313
200002
(200004,0)
```

 @王翔宇Marshall
weibo.com/xiangyuEyz


```
F:\rustbin\ProgDesignI\week6shiyan\Assignment1>source.exe
1
4
200002 12234 1832001
200003 15002 1745201
200004 18745 1883410
200001 24834 1921313
200001
(0,3)
```

 @王翔宇Marshall
weibo.com/xiangyuEyz

```
F:\rustbin\ProgDesignI\week6shiyan\Assignment1>source.exe
1
4
200002 12234 1832001
200003 15002 1745201
200004 18745 1883410
200001 24834 1921313
200003
(200004,0)
```

 @王翔宇Marshall
weibo.com/xiangyuEyz

```
F:\rustbin\ProgDesignI\week6shiyan\Assignment1>source.exe
1
4
200002 12234 1832001
200003 15002 1745201
200004 18745 1883410
200001 24834 1921313
200004
(200001,2)
```

 @王翔宇Marshall
weibo.com/xiangyuEyz

以及按原流程图缩写程序的测例：

```
F:\rustbin\ProgDesignI>TallTree_initial.exe
2
3
123456 14323 1700000
123458 41412 1900000
123457 15221 1800000
123458
(0,0)
4
200002 12234 1832001
200003 15002 1745201
200004 18745 1883410
200001 24834 1921313
200002
(200004,0)
```



@王翔宇Marshall
weibo.com/xiangyuEyz

```
F:\rustbin\ProgDesignI>TallTree_initial.exe
1
4
200002 12234 1832001
200003 15002 1745201
200004 18745 1883410
200001 24834 1921313
200001
(0,0)
```



@王翔宇Marshall
weibo.com/xiangyuEyz

显然，当查询最大值/包含分支节点作为子节点的元素时，输出错误。

Assignment 2

：本题为给定一个C程序，画出其流程图并写出伪代码。

个人感觉这道题主要考察的是细心和认真程度。源代码括号和缩进的嵌套十分多，极易被误解，翻译的过程中额外细心。没有感觉到其他难点。

不过一个好的文本编辑器（我也在用sublime Text 3）通常可以起到很大的作用。比如帮助配对每个括号。

Assignment Ex

我起初是把option理解成了section，于是两个都做了……既然都做了，我便把两个都提交好了。

option 1

判定回文素数。

对算法性能没有要求的话，本题的关键或许就是在大数字和字符串之间的转化。从师兄后来在群里临时增加的禁用所有与数值字符串互转有关的函数来看，也许确是这样。

在看到额外通知之后我的源码、伪码和流程图都已完成，又得知这次可以不改之后……我也决定不改了。

但其实即便没有这类函数，手写一个也不麻烦：

1. %s读入字符串（内容显然是一个数字）
2. strlen()得到字符串长度，然后分别取每一位字符，将其转变成数字，乘以对应的阶，结果累加存在一个long int中。
3. 对于回文数判定，手动实现的话，只需将下标从0遍历到 $(\text{int})(\text{lenth} - 1) / 2$ ，判定每一个str[i]与str[length - i - 1]是否相等即可
4. 素数判定，这里用了竞赛题中最直接想到的优化：将i的遍历上限从(n - 1)改为 $(\text{int})(\text{sqrt}(n + 1))$ 。
5. 两个问题：若是判定素数还能优化吗？若不能继续优化，本题若想拿满分是不是要从细节方面考虑？（比如说先算回文数，判定结果不是则直接结束，结果为No，从而节省时间）

option 2

：简化Assignment 2的源代码。

原版代码的第一步，我最开始认为没有用，所以我最开始的做法是，直接全部删掉，然后任意保留一个前进/转体模块。

重新读题才发现，机器人是要贴墙走的。最开始对wall的初始，其实是判定机器人是左手边贴墙还是右手边贴墙。那么两个差别不大的前进/转体模块便都不可被删去（左右手边贴墙的直接结果是机器人转体时的优先方向不同，源代码中写得确很清楚）。

所以作为优化，我剔除了wall变量，在一开始确定了贴墙方式之后，使其直接进入前进/转体模块。

Discussion , Cooperation and Summary

option 1 : Github Discussion



分享了scanf格式控制字串的一些细节问题。

option 2 : Simple Presentation

已向学委申请，打算在课上继续讲scanf以及数据读入的细节问题

option 3 : Helping Others

我帮助他人解决问题一般分为三个阶段：

1. 先听他讲，找出大概的问题，然后从他的思路顺延，从而容易发现问题的关键
2. 给一定的提示，引导他一起去想
3. 如果他想到答案了，自然最好，如果实在没想到，我就会考虑顺着思路说出我的答案。然后再总结。

更新的新手（虽然我也是新手）做题遇到困难大多都是因为了解的方法不够多。但一般都会有较为麻烦的替代方法。因此我一般会在帮助他们的时候，在中间或最后告诉他一个函数/常见写法。

比如本次wk6课堂题目的1004，奖学金那一道。许多同学卡在这里很久，我的两个舍友从开始做到最终做出来这道题，前后经过了一天多的时间（不是一直在想）。

我发现他们主要都是两个问题：

1. 都是在数据读入的时候出了问题，即录入数据的时候没有考虑%c会同样录入一个空格。
2. 数据存储没有思路（当时他们不了解二维数组）

第一个问题，我先告诉%c会录入空格，让他们换一个思路；后来我告诉他们可以用char[2]定义一个字符串（也是我后来在github上分享的那个内容，见option 1图）然后通过%s来录入，就可以避免这类问题；

第二个问题，他们最开始用了各种开一维数组的方式，但总是混乱。我给他们简单讲了一下二维数组是什么东西，然后鼓励他们自己查一下数组元素的引用，之后我稍一说，他们就很清楚该怎么做，很快A出了正确答案。

~之后又一个同学问到我%c录入空格的问题，他在scanf的控制字符串里加了空格之后，发现解决了这个问题，就来问我。我起初对此了解也不多，查了一下了解了这里的规定，给了他答复，自己也学习了更多。

option 4 : Personal Summary

来到大学之后终于开始了正规的编程学习，首先很高兴；不过虽然喜欢，也虽然有一点基础，但毕竟还是菜鸟，实际过程中也确实遇到了一些问题。

问题主要是熟练度方面，大致分为三个问题：

我认识一些优秀的同龄人，他们除了会做一些算法题之外，许多不要求太多算法，主要是模拟过程的题也可以做的很顺畅，显然是因为代码写得足够熟练&对所使用的语言的各项特性（尤其是一些高级特性）和擅长的工作内容了解得非常透彻。显然每个学生都想要成为这样的人，由此即是引申到的几个问题：

1. 就C语言来讲，哪些函数库最好优先了解？比如stdlib.h应该是一个很常用的库。
2. 如果想要比较好的了解C语言各类基本概念，我需要做很多实验。以算到期末考为准，我每天（或每周）大概要拿出高度专注的多久来专心尝试比较好？因为要学很多科目，我想对每科有一个规划。
3. （C语言最大的亮点特性是指针吗？）C语言适合做哪类工作？比如用python写网页相对就要轻松一些。

说到我学习/解决问题的方式，一般是先从问题开始。也是大致分三个阶段。

1. 寻找一个问题，先思考用已有的知识能不能解决（或很快的解决），若不能就去搜索有关内容，直到有了大概的了解。
2. 然后尝试独自写出问题的解决方案。在独自实现的过程中其实也必然会遇到很多细节问题，在这个过程中继续尝试与搜索的，直到完成这个实现。
3. 最后再回头继续思考直到完成也没能解决的问题，可能会反复多次实验。
问题是无限的，身边和电脑里无时无刻都有值得思考的地方，所以不存在找不到问题的情况。

Self Grading

所有模块的总分为 $20+35+35+10+15=115$ 分。

我给自己的估计的较高分数为 $20+30+32+8+10=100$ 分

一般分数为 $17+25+28+7+10=87$ 分。

较高分数指若评分没有我想象中的那么严苛时的标准

一般分数的估计原因则是当评价比较严苛的情况。

理由：

1. 我完成了所有的题目，在自己的测试中都看到了正确的（想要的）结果
2. 对于Assignment 1，我认为扣分的地方在于我的测试不够完善，测例不够多。也许我

自己的算法也存在问题，但我没有发现。

3. Assignment 2，这道题主要是翻译。考的是细心程度。我不敢完全保证我的每一个细节都是对的；另外也许还有一些老师/学长要求的东西我没有想到。
4. AssignmentEx：对于Option1，我本人能想到的扣分点在于没有提供测试过程。对于Option2，这道题我确实有做了一点simplify，但我总感觉可以继续简化（也许简化后计算量没有变，但流程图和代码可以更简洁）。不过没有想到一个比较合适的方式。
5. Discussion，Cooperation & Summary部分：因为我还是看到一些朋友很热衷帮助同学，所以我也不敢确定我是不是真正帮到其他人很多，应该会有比我更多的，所以我给自己打10/15分。

结语

这次实验题促使我们每个人对流程图和伪代码进行了一次系统的学习，获益的确很多。

debug的过程和查询的过程，前者让我写代码更加熟练&自信，后者让我了解了更多的东西。比如Process On，判定素数的算法，重写标准库函数，马克飞象可以直接将Markdown转PDF（本文即是如此写成），以及诸多细节问题。这篇文章保存下来，日后也可以作为我回顾的一个资料。