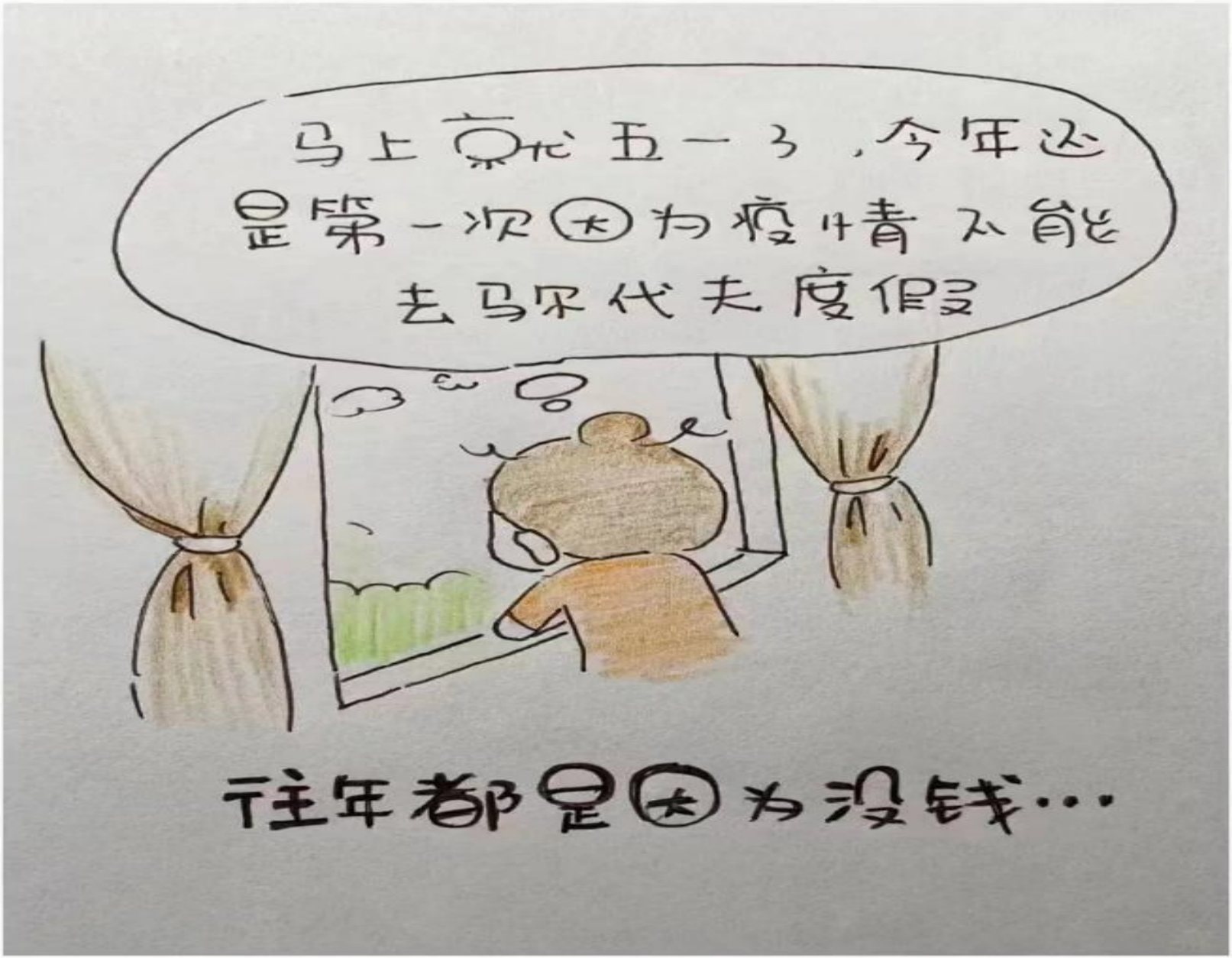
马上就五一了，今年还
是第一次因为疫情不能
去马尔代夫度假
往年都是因为没钱…

上海人民在封控中等物资
苏州人民在物资中等封控

苏州人民是最可爱的，自己决定封自己

Chapter 7 Writing the Programs

Daunting Task:

（令人气馁的任务）

① design is not always straightforward to coding

② coding should be understandable

③ considering reuse

④ check design

(聪明的程序员借此检查设计的诸多原则，同时也达到学习系统设计的目的)

Purpose(of the chapter):

① this chapter does not teach you how to program

② this chapter explains some of the software engineering practices

(guidelines or experience for implementation)

Chapter 7 Writing the Programs

7.1 Programming Standards and Procedures

(编程标准 (和步骤))

关于标准和规范问题，不止课件，华为也在强调：我们培养的不是编程高手，是软件工程师。不规范的文档无法通过验收。

focus on:

A: team work , many people involved

B: understand each other is important

C: organization's standards and procedures is important (about coding and for coder)

1. Standards for You（编程标准对自身的用处）

① organizing your thoughts and avoid mistakes

② keep tracking what we had been doing by documentation

Chapter 7 Writing the Programs

③ standards and procedure is helpful in translating designs to code

------it is easy to find that which or where the codes should be modified when we change the designs

2. Standards for Others（编程标准对他人的用处）

① easy to maintenance (example: change requirements)

② easy to testing(independent test team know how/what)

③ easy to reuse(by other separate team)

④ example: opening section (§7.1.2)

------explain the functions and interfaces invocations, coefficients, formula, return value, etc.

不能维护的模块不允许存在

Chapter 7 Writing the Programs

3. Matching Design with Implementation

（设计与编程实现相匹配） （如何做到？）

------direct correspondence between the program design components and the standardized program code components is essential standard (or critical standard)

------design characteristics ,such as low coupling, high cohesion, and well-defined interfaces, should also be program characteristics

（例如：我们很容易写一段不容易维护的代码，而完全忽视许多的设计原则。那将是不被允许的。）

Chapter 7 Writing the Programs

7.2 Programming Guidelines (编程的指导原则)

note:① 编程不仅仅是将设计转化为代码,而是有着很大的灵活性和创造性

② the section is not language-specific guidelines(特定语言指南)

③ general programming guideline(一般性编程指导原则)

④ 同样一段程序代码，不同程序员可以写出完全不同的性价比。

component include: A: control structure（控制结构）

B: algorithms（算法）

C: data structure（数据结构）

1. Control Structures

note:A: the highest programming guideline----read a component easily (the coders should concentrate on what is being done, not on the control flow)

B: in implicit invocation or OO design, control is based on the system states and changes in variables.

团队软件工程的管理目标

Chapter 7 Writing the Programs

In more procedure designs, control depends on the structure of the code itself.

但不管什么样的设计，都要使程序结构反映设计的控制结构

① restructuring can aid understanding
example—rearranging codes (P377)

② modularity makes coding understandable (through hiding details using macros, procedures, subroutines, methods and inheritance)

③ generality: make control structures be in more general (generality is a virtue) （代码不可太特殊）

④ coupling (among components) must be visible
example— (P378)
(即部件之间的耦合或依赖关系必须是可见的)

Chapter 7 Writing the Programs

- Control skips around among the program's statements

```
            benefit = minimum;
            if (age < 75) goto A;
            benefit = maximum;
            goto C;
            if (AGE < 65) goto B;
            if (AGE < 55) goto C;
A:          if (AGE < 65) goto B;
            benefit = benefit * 1.5 + bonus;
            goto C;
B:                  if (age < 55) goto C;
            benefit = benefit * 1.5;
C:          next statement
```

- Rearrange the code

```
if (age < 55) benefit = minimum;
elseif (AGE < 65) benefit = minimum + bonus;
elseif (AGE < 75) benefit = minimum * 1.5 + bonus;
else benefit = maximum;
```

Chapter 7 Writing the Programs

2. Algorithms

----the coders have great deal of flexibility in converting the algorithm to code.

① pursuing efficiency may have hidden cost

(追求效率可能有潜在成本：编写更快代码的代价，测试代码的时间代价，用户理解代码的时间代价等等)

example----(P378--4 dots)

② pursuing efficiency may sacrifice clarity and correctness （一般情况下，清晰度第一，效率第二。）

③ learning how to optimizing codes by compiler

example----compute index of an array

Chapter 7 Writing the Programs

3. Data Structures

① keeping the program simple (简化程序(通过改变模块DS))

----restructuring data can simplify a program's calculation

----example (calculate the federal income tax due) (P379-380)

② using a data structure to determine a program structure

A: data structure $\xrightarrow{\text{determine}}$ program structure

B: data structure $\xrightarrow{\text{determine}}$ the choice of language

example----a recursive defined tree

Example: Determining Federal Income Tax

1. For the first $10,000 of income, the tax is 10%
2. For the next $10,000 of income above $10,000, the tax is 12 percent
3. For the next $10,000 of income above $20,000, the tax is 15 percent
4. For the next $10,000 of income above $30,000, the tax is 18 percent
5. For any income above $40,000, the tax is 20 percent

```
tax = 0.
if (taxable_income == 0) goto EXIT;
if (taxable_income > 10000) tax = tax + 1000;
else{
        tax = tax + .10*taxable_income;
        goto EXIT;
}
if (taxable_income > 20000) tax = tax + 1200;
else{
        tax = tax + .12*(taxable_income-10000):
        goto EXIT;
}
if (taxable_income > 30000) tax = tax + 1500;
else{
        tax = tax + .15*(taxable_income-20000);
        goto EXIT;
}
if (taxable_income < 40000){
        tax = tax + .18*(taxable_income-30000);
        goto EXIT;
}
else  tax = tax + 1800. + .20*(taxable_income-40000);
EXIT:
```

7.2 Programming Guidelines
Keep the Program Simple Example (continued)

- Define a tax table for each “bracket” of tax liability

Bracket	Base	Percent
0	0	10
10,000	1000	12
20,000	2200	15
30,000	3700	18
40,000	55000	20

- Simplified algorithm

```
for (int i=2; level=1; i <= 5; i++)
        if (taxable_icome > bracket[i])
                level = level + 1;
tax= base[level]+percent[level] * (taxable_income -
 bracket[level]);
```

Chapter 7 Writing the Programs

4. General Guidelines/strategies(其他通用编程策略)

① localizing input and output (局部化输入输出 / 单独设计I/O)

----making maintenance more easily

② including Pseudocode (设计阶段包含伪代码及其改进)

A: focus on:

Pseudocode —creativity→ source code(have most desirable structure)

—expertise→

B: example—text process system （有时详细设计太粗）

a: Pseudocode (in program design stage) (P382-385)

b: list intermediate Pseudocode (subactions)(P383)

Chapter 7 Writing the Programs

c: regroup the common sub-actions(P383-384)

d: improvement (P384)

e: final design document (P384-385)

C: note: it is possible and necessary to change program design

③ revising and rewriting, not patching

（改动时从需求改动, 重新设计、重新编码, 不要打补丁）

④ reuse（重用）

A: two kind of reuse:

X: producer reuse（生产者自重用）

Y: customer reuse（外部用户重用）

Chapter 7 Writing the Programs

B: four key characteristics (about consumer reuse)

----1-4 characteristics(P386) (1.良好的功能 2.易修改性 3.文档化水平 4.测试记录)

C: several things (in producer reuse)

----1-7 dots(P386)

7.2 Programming Guidelines
Example of Pseudocode

The design for a component of a text processing system states（本阶段不太关心实现，只大致罗列各功能模块）

```
COMPONENT PARSE_LINE
  Read nest eighty characters.
      IF this is a continuation of the previous line,
          • Call CONTINUE
      ELSE determine command type
      ENDIF
  CASE of COMMAND_TYPE
      COMMAND_TYPE is paragraph: Call PARAGRAPH (段落操作)
      COMMAND_TYPE is indent : Call INDENT （缩进）
      COMMAND_TYPE is skip line: Call SKIP_LINE （跳行）
      COMMAND_TYPE is margin : Call MARGIN （页边距）
      COMMAND_TYPE is new page : Call PAGE （分页）
      COMMAND_TYPE is double space : Call DOUBLE_SPACE （双倍行间距）
      COMMAND_TYPE is single space : Call SINGLE_SPACE
      COMMAND_TYPE is break : Call BREAK （断行）
      COMMAND_TYPE is anything else: Call ERROR （错误处理）
  ENDCASE
```

7.2 Programming Guidelines
Example of Pseudocode (continued)

- Intermediate pseudocode（细化成各个子动作）

PARAGRAPH:

Break line, flush line buffer. Advance one line between paragraph. If fewer than 2 line left on page, eject. Set line pointer to paragraph indent.

INDENT:

Break line, flush line buffer. Get indent parameter. Set line pointer to indent parameter, set left margin to indent.

SKIP_LINE:

Break line, flush line buffer. Get line parameter. Advance (parameter) lines or eject if not enough space left on current page.

MARGIN:

Break line, flush line buffer. Get margin parameter. Set line pointer to left margin. Set right margin to margin.

PAGE:

Break line, flush line buffer. Eject page. Set line pointer to left margin

SOUBLE_SPACE:

Set interline space to 2.

SINGLE_SPACE:

Set interline space to 1

BREAK:

Break line, flush line buffer. Set pointer to left margin

7.2 Programming Guidelines
Example of Pseudocode (continued)

- Regrouped（整理出各个公共子动作组）

FIRST:

PARAGRAPH, INDENT, SKIP_LINE, MARGIN, BREAK, PAGE:
Break line, flush line buffer.
SOUBLE_SPACE, SINGLE_SPACE :
No break line, no flush line buffer.

SECOND:

INDENT, SKIP_LINE, MARGIN:
Get parameter.
PARAGRAPH, BREAK, PAGE, DOUBLE_SPACE, SINGLE_SPACE:
No parameter needed.

THIRD:

PARAGRAPH, INDENT, SKIP_LINE, MARGIN, BREAK, PAGE:
Set new line pointer.
DOUBLE_SPACE, SINGLE_SPACE:
New line pointer unchanged.

FOURTH:

Individual action taken

7.2 Programming Guidelines
Example of Pseudocode (continued)

- Final pseudocode（根据子动作的依赖关系进行进一步改进）

```
INITIAL:
        Get parameter for indent, skip_line, margin.
        Set left margin to parameter for indent.
        Set temporary line pointer to left margin for all but paragraph;
          for paragraph, set to paragraph indent.
LINE_BREAKS:
        If not (DOUBLE_SPACE or SINGLE_SPACE), break line, flush
          line buffer and set line pointer to temporary line pointer
        If 0 lines left on page, eject page and print page header.
INDIVIDUAL CASES:
        INDENT, BREAK: do nothing.
        SKIP_LINE: skip parameter lines or eject
        PARAGRAPH: advance 1 line; if < 2 lines or page, eject.
        MARGIN: right_margin = parameter.
        DOUBLE_SPACE: interline_space = 2.
        SINGLE_SPACE: interline_space = 1;
        PAGE: eject page, print page header
```

注：最后一段没有展示

Chapter 7 Writing the Programs

7.3 Documentation（文档化）(in coding stage)

program documentation: a written description, explain "what" and "how" , which includes:

- internal documentation: description material (within the source codes)
- external documentation: all other documentation

Chapter 7 Writing the Programs

1. Internal documentation（内部文档）

note: comment information for source codes reader.
Include header comment and other program comments.

① header comment block （头部注释版块）(HCB)

A: definition: the summary information (used to identify the program, and describe data structure, algorithms, control flow)

B: explaining of HCB (P387: 1-6 and text explaining)

C: detailed explaining (P387: 5 dots)

D: example of HCB (P388) （7.3.1节）

Chapter 7 Writing the Programs

② other program comments（其他程序注释）

A: explaining: other explaining (exclude HCB) to help readers understand all intentions about source codes.

B: simple guidelines

note: additional comments are useful although structured code

u: phased comments (exclude lined comments)

v: code change accompanying comment update

（一定要养成这个习惯：修改代码与修改注释相伴而行）

w: comments should have new information

example—(P388)

x: writing comments as writing code: not afterward

Chapter 7 Writing the Programs

③ meaningful variable names and statement labels

A: express specific meaning or useness

B: it is better alphabetic statement labels

④ formatting to enhance understanding (格式化问题)

A: indentation and spacing

----clarity and formatting (of the source code)

B: right comments

⑤ documenting data（数据文档化/记录数据）

A: internal document should include description for DS and its useness

B: information hiding in OO make it even more difficult to understand how a data value is changed

Chapter 7 Writing the Programs

2. External Documentation(外部文档)

Note:

A: internal document is for programmer

external document is for those who never read codes (for example: designer will taking modification or enhancement); it answers questions in a system view. （如：奉命修改某算法，则需要调查该算法的出处，则需要这样的外部文档说明）

外部文档类似于拓展阅读

B: content: X: overview （概述）

Y: data sharing and using （如何共享）

Z: explain object classes and inheritance hierarchy （类与继承层次的实现目的）

Chapter 7 Writing the Programs

C: different with the design documentation:

design document----skeleton

external document----flesh / muscle

① describing the problem

A: why a particular solution was chosen

B: discussing the background of the problem

② describing the algorithms

focus: where, formula, boundary condition

----supplement explaining about algorithm in design or other document.

Chapter 7 Writing the Programs

③ describing the data

A: data flow description in model level

B: explaining the interaction among objects in OO components.

(interdependency, dealing sequence, constraints, etc)

Chapter 7 Writing the Programs

7.4 The Programming Process(编程过程)

Note: guidelines(指导原则) of programming process

1. Programming as Problem Solving

(将编程作为问题求解过程)（理解与学习的过程）

four stages：

(1) Understanding the problem(nature about a problem)

(2) Devising a plan (solution)

(3) Carrying out the plan

(finish the solution and implementation)

(4) Looking back (回顾----check, modify the implementation)

Chapter 7 Writing the Programs

2.Extreme Programming

极限编程（XP）是一种轻量级的软件开发方法论，属于敏捷开发方法。XP从实践中来，是对实践的总结，也是经过实践检验的，其主要特征是要适应环境变化和需求变化，充分发挥开发人员的主动精神。XP承诺降低软件项目风险，改善业务变化的反应能力，提高开发期间的生产力，为软件开发过程增加乐趣等等 。

两类参与者:

客户: 定义将要实现的系统之特征; 描述测试计划;
分配系统实现和测试的优先级

程序员: 将客户的上述诉求予以编程实现

Chapter 7 Writing the Programs

3.Pair Programming

派对编程属于主要的敏捷开发方法，其开发方式是两个程序员共同开发程序，且角色分工明确。一个负责编写程序，另一个负责复审与测试。两人定期交换角色。

(思考: 敏捷开发方法的问题或弱点可能会出在哪里?)

4.Whither Programming ? (编程向何处去?)

面对大型的,关键任务的软件: 需要对极限编程添加额外的步骤(文档化思想----定义基线体系结构, 使用文档化场景,定义系统边界等等)（产业界敏捷，实际上提取了重量级方法的若干要素）

5.课程设计与班级练习项目所采用的过程或方法

----可能比敏捷方法复杂些, 远低于重量级方法.

Chapter 7 Writing the Programs

7.5 The Code Review (代码复审) （补充内容）

代码复审是在代码编写完成后，由专门的复审人员参照编码规范对源代码重新进行阅读和检查的过程。构建高质量的软件时，代码复审是对编译、集成和测试等其他质量保证机制的补充，在复审代码之前，要对其进行编译，并使用诸如代码规则检查器之类的工具发现尽可能多的错误。如果使用可在运行时检测错误的工具来执行代码，还能够在进行代码复审之前检测和消除其他错误

1.复审代码的意义（大体上的意义和要求）

（1） 加强和鼓励在项目中使用一种共同的编码风格。复审代码是一种有效的让成员遵循编程指南的方法。要确保成员遵循编程指南，复审所有作者和实施者的工作结果比复审所有源代码文件重要得多。（编码阶段，只审核一部分内容吧，有时是看表面，有时要细查。）

Chapter 7 Writing the Programs

（2） 发现自动测试发现不了的错误。代码复审捕捉到的错误与测试发现的错误不同，利于交流个人知识，让富有经验的个人将知识传授给经验较少的新手。（合理性/有效性）

2.复审代码的方法。

（1）检查。(正式)一种用来详细检查代码的正式评估方法。尽管需要培训和准备，但检查仍被认为是最有效复审方法。

（2）走查。(走马观花)一种由代码制作者带领一个或多个复审员遍历代码的评估方法。复审员针对技术、风格、可能出现的错误、是否违反编码标准等提出问题，并发表意见。

（3）阅读代码。由一两个人阅读代码，如果复审员阅读完毕，他们可以碰头并提出各自的问题和意见。尽管碰头也可以省略，但复审员可以通过书面形式向代码制作者提出他们的问题和意见。在对一些小的改动进行检验时，建议使用阅读代码方法，它是一种“正常性检查”。（中等认真）

Chapter 7 Writing the Programs

3、源代码检查要点

本部分给出了一些通用的代码复审检查要点，仅作为复审查找对象的示例，编程指南应作为代码质量的主要信息源。

(1)概述

A: 代码是否遵循编程规范?

B: 通过阅读能否较好地理解代码?

C: 是否已经解决了代码规则检查和(或)运行错误检测工具发现的所有错误?

(2)注释

A: 注释是否反映了最新的修改情况?

B: 注释是否清晰、正确?

C: 如果代码被变更，修改注释是否容易?

D: 注释是否着重解释了为什么，而不是怎么样?

E: 是否所有的意外、异常情况和解决方法错误都有注释?

Chapter 7 Writing the Programs

F: 每个操作的目的是否都有注释?

G:与每个操作有关的其他事实是否都有注释?

(3)源代码

A: 是否每一个操作都有一个描述其操作内容的名称?

B: 参数是否有描述性的名称?

C: 完成各个操作的正常路径是否与其他异常路径有明显区别?

D: 操作是否太长，它能否通过将有关语句提取到专用操作中进行简化?

E: 操作是否太长，它能否通过减少判定点的数目进行简化?决策点是代码可以采取不同路径的语句，例如，**if**、**else**、**while** 和 **switch** 语句。

F: 循环嵌套是否已减至最少?

G: 变量命名是否适当?

H:代码是否简单明了，是否过度使用“技巧”？

Chapter 9 Testing the System

Note A:unit and integration testing----by yourself or a small part of the development team

B:system testing----by the entire develop team

- **9.1 Principles of system testing**

Focus A: ① objective of unit and integration

------ensure the code implemented the design properly ①②③④⑤⑥ ⑦ ⑧⑨⑩