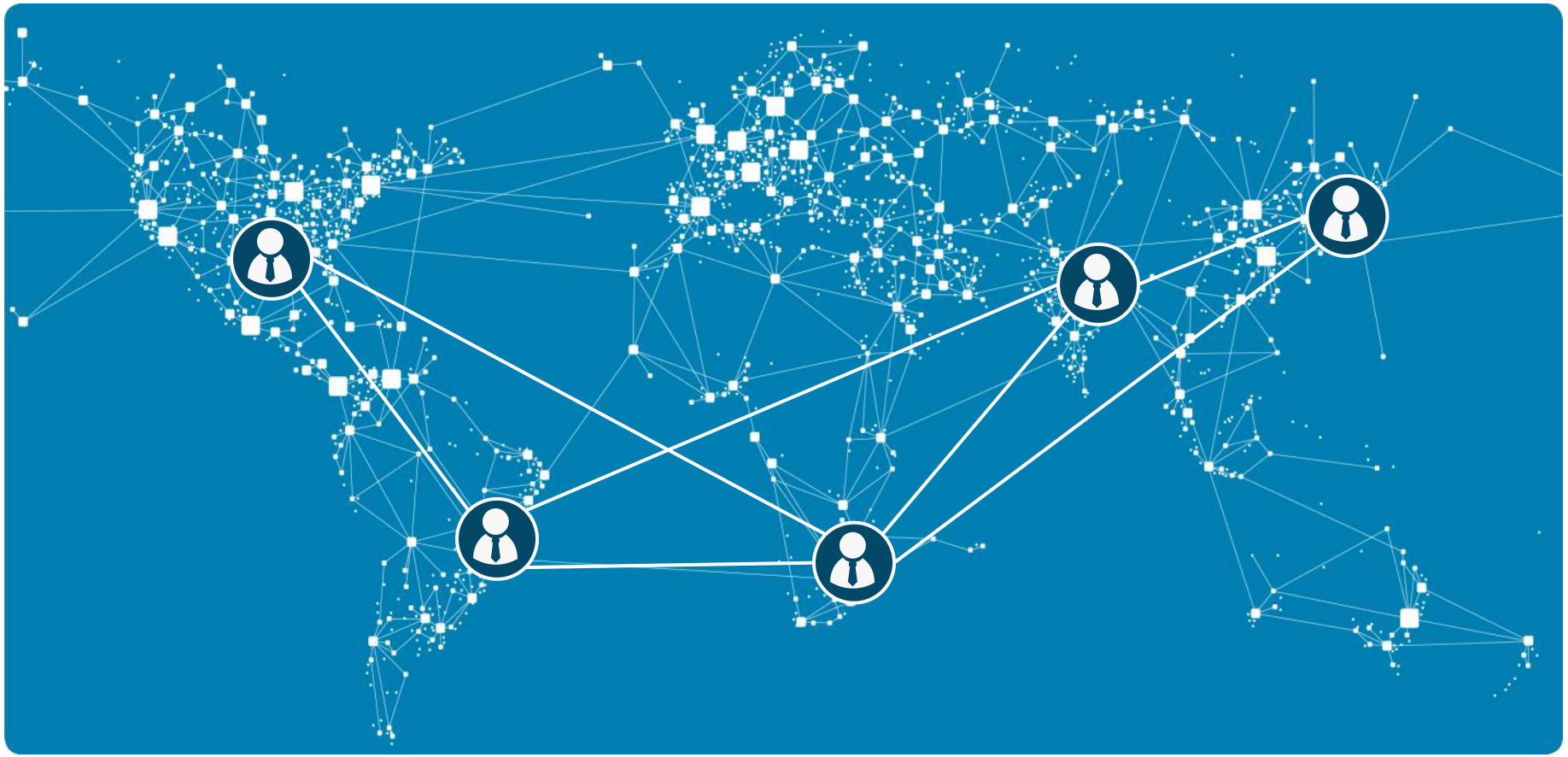


实验一：结对编程

实验目标

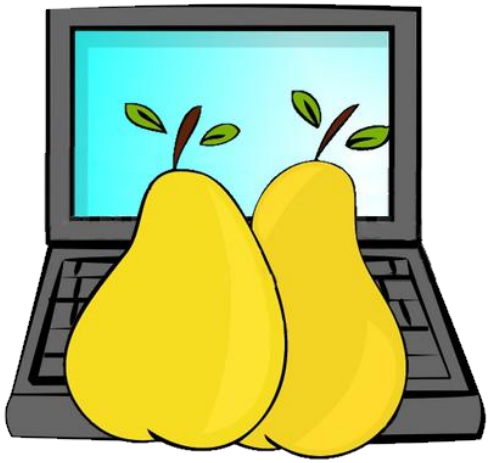
- 练习结对编程，体验敏捷开发中的两人合作

- 两人一组，自由组合；
- 使用一台计算机，共同编码，完成实验要求；
- 在工作期间，两人的角色至少切换4次；
- 建议使用Python或JAVA进行编程。

- 实验要求：编程实现生命游戏或俄罗斯方块等，给出设计算法及实验结果。

结对编程（Pair programming）

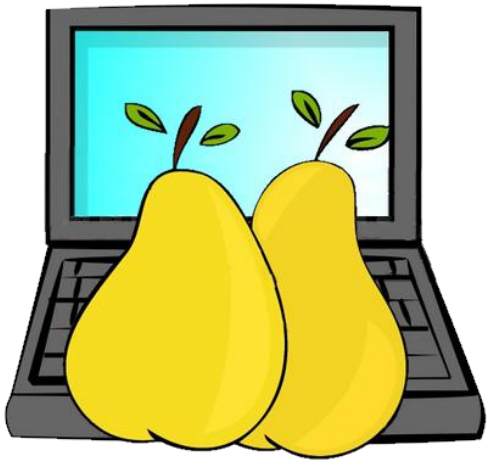
- Two developers writing code at a single workstation with
  - only one typing
  - continuous free-form discussion and review



Introduction to Pair Programming,version 2

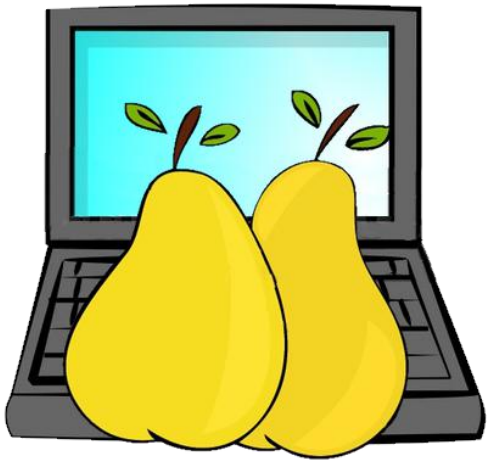
结对编程（Pair programming）

- Two developers writing code at a single workstation with
  - only one typing
  - continuous free-form discussion and review
- Advantages
  - Deep reviews, instant and continuous feedback.
  - Learning, sharing, team-building.



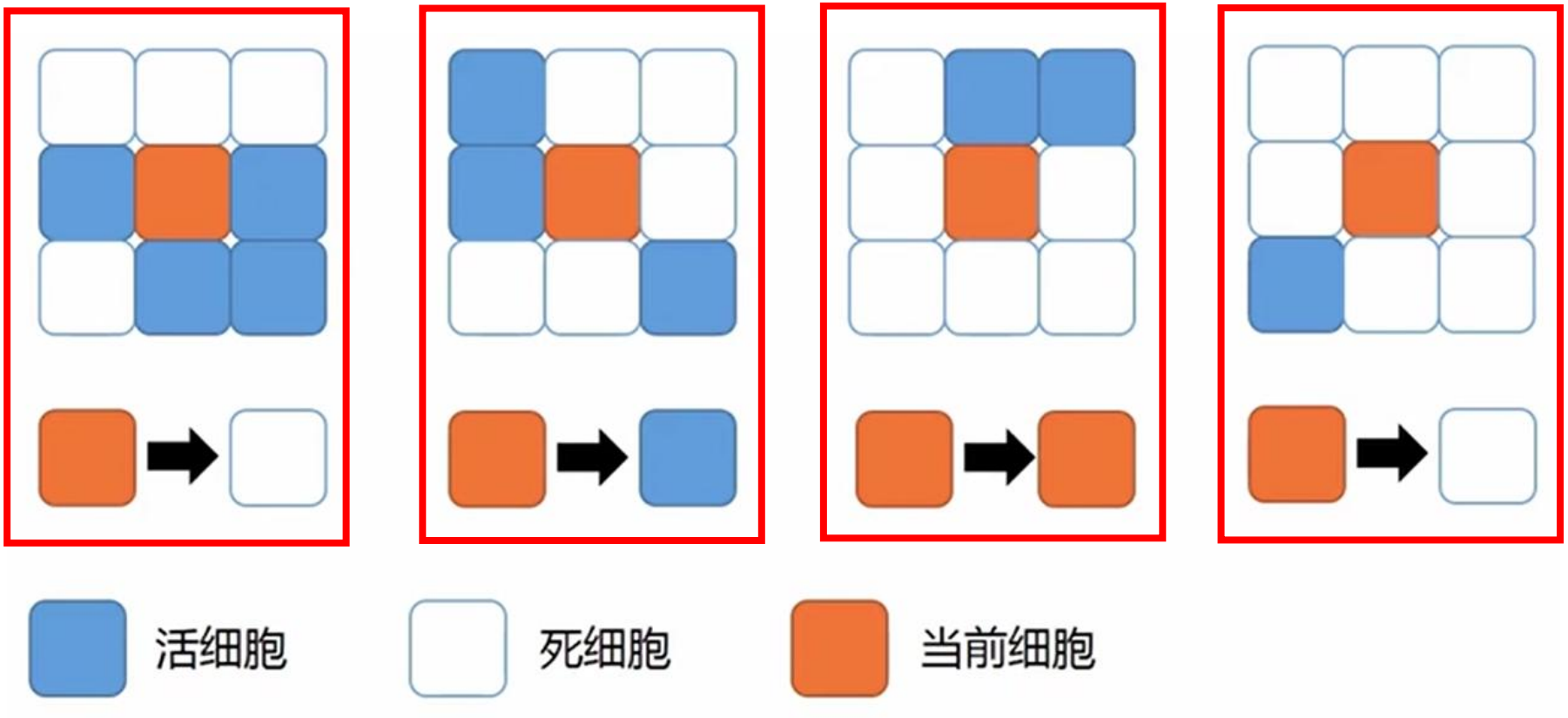
结对编程（Pair programming）

- Two developers writing code at a single workstation with
  - only one typing
  - continuous free-form discussion and review
- Advantages
  - Deep reviews, instant and continuous feedback.
  - Learning, sharing, team-building.
- Disadvantages
  - Some developers don't like it.
  - No record of the review process.
  - Time consuming.



问题描述

生命游戏是英国数学家约翰·何顿·康威在1970年发明的细胞自动机。



模块的划分

地图

管理与地图相关一切数据的初始化、获取、更新等

控制完整游戏逻辑，根据地图数据依照逻辑进行相应更新

负责时间相关的功能，在适当的时机触发游戏逻辑模块对地图的更新

各模块接口设计

案例：生命游戏

地图

```python
# game_map.py
class GameMap(object):

    def __init__(self, rows, cols):
        """地图将在逻辑模块进行初始化"""
        pass

    def reset(self, life_ratio):
        """重置地图并按life_ratio随机地填充一些活细胞"""
        pass

    def get_neighbor_count(self, row, col):
        """地图上一个方格周围活细胞数是游戏逻辑里的重要数据"""
        pass

    def set(self, row, col, val):
        """当游戏进行中，需要常常更新地图上方格的状态"""
        pass

    def get(self, row, col):
        """当需要将游戏状态呈现给用户时，就需要获取地图上方格的状态"""
        pass
```

各模块接口设计

案例：生命游戏


```python
# life_game.py
class LifeGame(object):

    def __init__(self, map_rows, map_cols, life_init_ratio):
        """将在主程序中初始化实例"""
        pass

    def game_cycle(self):
        """
        进行一次游戏循环，将在此完成地图的更新
        将在计时器触发时被调用
        """
        pass

    def print_map(self):
        """由于暂时没有UI模块，因此先在逻辑模块进行地图的呈现"""
        pass
```

各模块接口设计

案例：生命游戏

计时

```
# game_timer.py
class GameTimer(object):

    def __init__(self, trigger, interval):
        """
        将在主程序中初始化实例
        计时器以interval秒的频率触发
        trigger是个函数，计时器被触发时调用该函数
        """
        pass

    def start(self):
        """启动计时器，之后将以interval秒的间隔持续触发"""
        pass
```

实验报告

- 提交截止时间：下周实验课前一天
- 提交内容（打包后提交给各班学委）：
  - 实验报告：　命名规则“班级-学号-姓名-Lab1-report.doc”
  - 程序源代码：命名规则“班级-学号-姓名-Lab1-code.java”

注意：实验报告内容需包含以下：

1. 实验要求
2. 问题描述
3. 算法设计思路
4. 实验结果
5. 结对编程过程
   - 5.1 角色互换时间点以及各自的任务分工
   - 5.2 工作照片
6. 结对编程之体会

Project I - Elevator Simulation System

Project Descriptions:

The project requires each team use Java to develop an elevator simulation system (called ESS)that simulates a real elevator control system to support a 10-floor building with an elevator. The system is host-centered program supporting two types of users:

- End-users - Who access the elevator system by accessing Floor Operation Panels and Internal Operation Panel in the elevator.
- System administrators - Who access the system Admin Panel to maintain and check elevators.

You are to simulate the movements of an elevator, which works according to the following rules:

1) The elevator stays 3 seconds for customers to get in or get out. If some customers want to get in and some want to get out, the elevator stays 6 seconds.

2) When there is no request for service, the elevator stays where it is. Namely, it is idle.

3) When the elevator is idle, and a request comes from upwards or downwards, it moves upwards or downwards. If a request comes from the story where it stays, the elevator opens its door to serve these customers.

4) When the elevator goes upwards or downwards, it keeps its direction until there is no request from upwards or downwards. However, it breaks off to let customers get out or pick up customers if necessary.

Project I - Elevator Simulation System

5) When the elevator can act more than one way according to aforementioned rules, it chooses the action with the highest priority. The priorities for actions, from the highest to the lowest, are:

let customers get out
let customers get in
go downstairs
go upstairs

The ESS system consists of the following parts:

- Central Controller - which starts, maintains, and monitors the elevator.
- The elevator system - which consists of
  - o 10 Floor-Panels each of which has two buttons (UP/Down).
  - o 1 Internal Operation Panel to support end-user accesses and send service requests to Elevator controller. Each panel includes the following operation buttons: OPEN, CLOSE, EMERGENCY, STOP, Floor-Numbers (from 1 to 10).
  - o 10 floor-indicators which display the current location of the elevator.
  - o Service processor which receives and processes user service requests.
  - o Elevator controller which controls the elevator's operations, such as Move-up,Move-Down, Open-Door, Close-Door, and Stop.
- A system admin operation user interface, which supports system

Project I - Elevator Simulation System

administrators to access the following functions:

- o Select, search, and monitor the elevator' status.
- o Receive and display the EMERGENCY requests.
- o Generate and display a status report for user services for the elevator.

The ESS system has one data repository that stores the following types of information:

- System admin records - which stores the user account and password for each administrator.
- Elevator status records - Each includes the elevator's service status, elevator ID, and current floors, etc.
- Elevator control records - Each includes the elevator's control information, such as elevator ID, name, deployment time, motor service status, and so on.
- User floor service request records - Each includes the required information for each type of requests for each floor.

The ESS system needs the following simulation user interfaces:

- For the elevator, you need one Internal Operation Panel.
- For each floor, you need one Floor Operation Panel.
- For the elevator, you need one simulation interface to show various types of the elevator statuses.
- One user interface to support system administration functions, such as status monitoring and reporting.

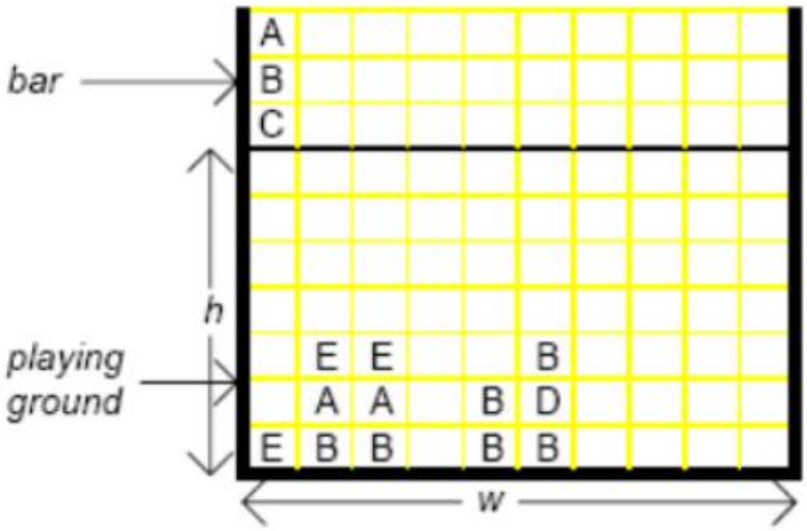
In the user manual, you must provide the following information:

- All screen layouts and input/output formats.
- All necessary operational information for end-users and system administrators.

Project II - Columns Game

***Project Descriptions*:**

The game *Columns* is rather well known, at least in its dynamic form, as TETRIS. The player can manipulate a bar of colored squares while it drops down. Once it has reached the playing field, points are gained and squares in the playing field disappear, according to certain rules. This game is very exciting, as the bars will drop down faster and faster as the game goes on. For some people it turns out to be too exciting, as they get a nervous breakdown from playing this game. For those people the static variant was invented. (Of course not playing at all would be a better idea. But the real addicts get a nervous breakdown if they don't play as well.)



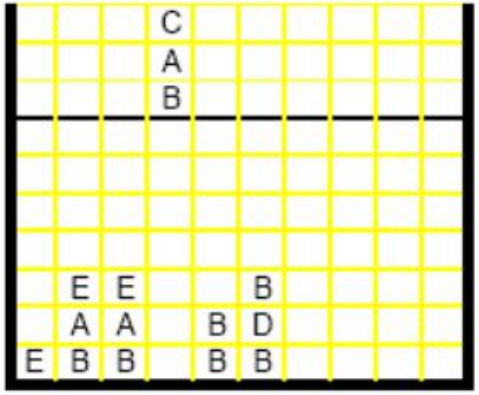
Let me explain *Static Columns* to you. The playing field is a rectangle of width w and height h. Initially it is empty, but after some time it may be filled with colored squares. In the scheme below the colors are indicated with capitals. The squares are subject to gravity: a square rests either on the floor or on another square. Repeatedly a vertical bar (extent n, now n=1) of colored squares appears on top of the playing field, against the left wall.

Project II - Columns Game

The player may manipulate this bar using (repeatedly) any of the following commands:

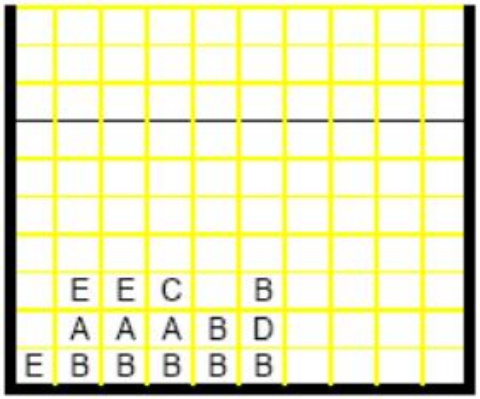
Command	Action
r (right)	Move the bar one step to the right. This command has no effect if the bar is already at the rightmost position of the playing field.
l (left)	Move the bar one step to the left. This command has no effect if the bar is already at the leftmost position of the playing field.
s (shift)	Shift (or actually rotate) the colors in the bar. The rotation is downwards: every color goes down one step, and the lowermost color is put on top.
d (drop)	Drop the bar. After this command, no other commands can be given for that bar.

After the commands "rrrs" the situation in the above example will be:

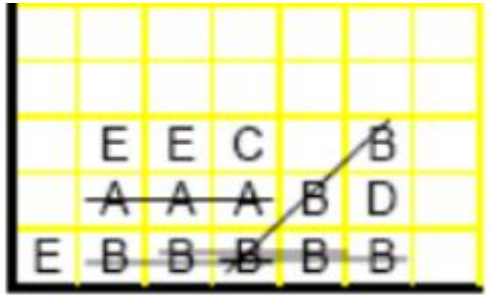


Project II - Columns Game

The drop command makes the bar going straight down until it is stopped, either by the floor or by a colored square.

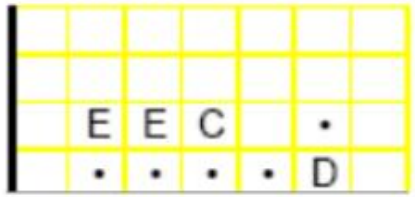


If the playing ground is very full, or the bar is very long it may happen that the bar is stopped before it is fully inside the playing ground. In that case the game is over and no other bars will be played. If not, two things happen. First, credits are counted, depending of the parameter l, the length of a series. Whenever l squares in the field in a series (horizontal, vertical or diagonal) have the same color, one point is given to the player. As indicated in the next diagram, the player receives 5 points if l=3 (the player would receive 2 points if l=4 and 1 points if l=5).

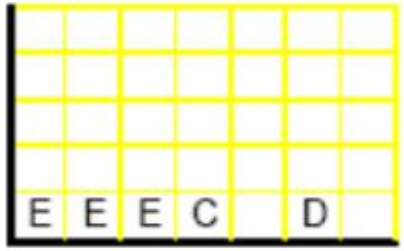


Project II - Columns Game

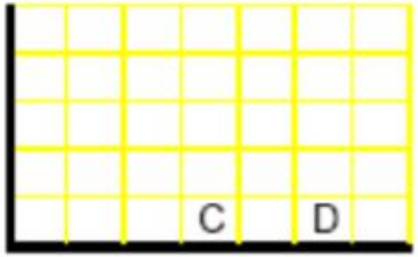
After all squares occurring in one or more such a series have disappeared, the following diagram is obtained:



Now gravity takes over, and the diagram becomes:



Once again a series is found and the player obtains one more point. The final situation is:



Using the Internet every move of every players is broadcasted all over the world. People with beautiful graphics on their machine will be able to replay the game.
You are ask to write a program to calculate the credits.