A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

12.2 Program Design

Name: _____ Class: _____ Date: _____ **Total: 28 marks**

KEY WORDS structure chart 结构图 • state-transition diagram 状态转换图 • parameters 参数 • states 状态 • program design 程序设计

Objective

Build the skills to answer exam questions on **program design** 程序设计 tools — structure charts and state-transition diagrams.

You must be able to:

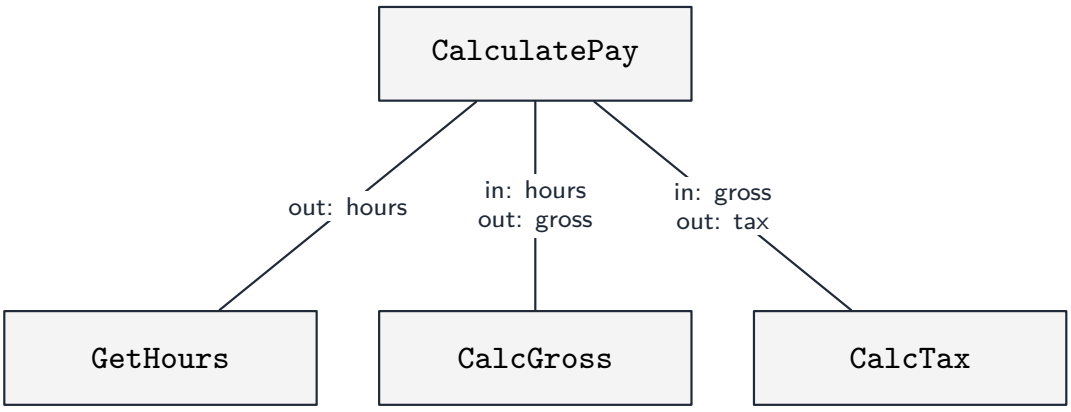
- read and draw a **structure chart** showing modules and the **parameters** between them
- read and draw a **state-transition diagram** showing states and events

1 Worked examples

Study these first. Each one shows the method for a question type used later — follow the steps and you can do the Practice and Exam-style questions yourself.

■ Structure chart

A **structure chart** shows how a program is **decomposed** into modules, and the **parameters** passed between them. Each module is a box; a line joins a caller (above) to a module it calls (below); the labels show data going **in** (down) and results coming **out** (up).

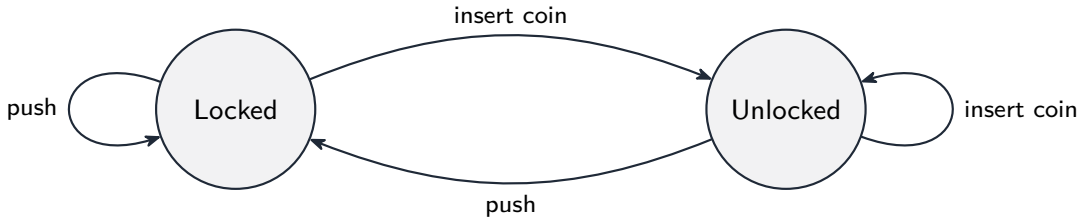


You can read each module's **signature** (its parameters and return) off the chart. Two

extra symbols may appear: a **curved arrow** over the links means those modules are called repeatedly in a **loop** (iteration); a small **diamond** on a link means the module is called only **sometimes** (selection).

■ State-transition diagram

A **state-transition diagram** shows the **states** a system can be in (circles) and the **events** (labelled arrows) that move between them. Below is a turnstile:



It makes **missing transitions** easy to spot — *"what happens if I push while Locked?"*

2 Practice

Now apply the methods above.

2.1 State what a **structure chart** shows. [1]

2.2 In a structure chart, what do the **labels on the links** represent? [1]

2.3 State what a **state-transition diagram** shows. [1]

2.4 In the turnstile diagram, which **event** moves it from `Locked` to `Unlocked`? [1]

3 Exam-style questions

3.1 Draw a **structure chart** for a program `MakeTea` that calls `BoilWater` and `AddMilk`. `AddMilk` is passed the number of `Cups`. Show the parameter on the link. [4]

3.2 A vending machine has states `Idle`, `Paid` and `Dispensing`. Draw a **state-transition diagram** with these events: `coin` (Idle→Paid), `select` (Paid→Dispensing), `done` (Dispensing→Idle). [4]

3.3 State **one** benefit of drawing a state-transition diagram. [1]

3.4 In a structure chart, state what a **curved arrow** drawn over the links to two modules means. [1]

3.5 A vending machine program is documented with a **state-transition diagram** and a **structure chart**.

(a) The machine has states `Idle`, `Collecting`, `Dispensing` and `OutOfStock`. **Complete** the state-transition table for the events shown. [4]

current state	event	next state
`Idle`	coin inserted	_____
`Collecting`	enough money and item available	_____
`Collecting`	enough money, item sold out	_____
`Dispensing`	item delivered	_____

(b) **State** what a state-transition diagram shows that a structure chart does **not**. [2]

(c) **Explain** the meaning of the **curved arrow** symbol in a structure chart. [2]

(d) In the chart, `ProcessSale` calls `CheckStock`, which returns a Boolean. **Describe** how the parameter and its return value are drawn. [3]

(e) The team also produces a **program flowchart** for one module. **Explain** why all three documents are produced, rather than choosing whichever one the team prefers. [3]

Solutions

2.1 The **hierarchical decomposition** of a program into **modules/subroutines** and the **parameters** passed between them.

2.2 The **parameters** passed **into** a module and the **results returned** from it.

2.3 The **states** a system can be in and the **events** (transitions) that move it between them.

2.4 `insert coin` (the coin event).

3.1 A correct chart: `MakeTea` at the top with lines down to `BoilWater` and `AddMilk`, with `Cups` shown as a parameter **in** on the `AddMilk` link.

3.2 Three states `Idle`, `Paid`, `Dispensing`, with arrows: `Idle →(coin)→ Paid`, `Paid →(select)→ Dispensing`, `Dispensing →(done)→ Idle` — each transition labelled with its event.

3.3 Any one: it makes **missing or invalid transitions** easy to spot; it documents the system's behaviour clearly for all events.

3.4 The two modules are called repeatedly in a **loop** — it shows **iteration**.

3.5 (a) `Idle` + coin inserted → `Collecting`; `Collecting` + enough money and item available → `Dispensing`; `Collecting` + sold out → `OutOfStock`; `Dispensing` + item delivered → `Idle`.
(b) It shows the **states** the system can be in and which **events** move it between them — the behaviour over time, including what is not allowed, which a structure chart cannot express because it only shows which module calls which.
(c) A curved arrow means **iteration**: the modules it encloses are called **repeatedly**, for as long as the condition on the loop holds. (A diamond, by contrast, marks selection.)
(d) Parameters are drawn as small arrows beside the line joining the two modules. The parameter passed **down** to `CheckStock` — the item code — points from `ProcessSale` towards it; the Boolean returned points **back up** towards `ProcessSale`, and each arrow is labelled with the data it carries.
(e) Each shows a different view: the structure chart shows how the program is **decomposed** into modules and what data passes between them; the state-transition diagram shows the **behaviour** of the system as a whole; the flowchart shows the **detailed logic inside one module**. No single one of the three carries all of that.