APPENDIX 2

Solutions to exercises

Chapter 1

1.

(a) Monday – clearly the Friday is the 27th.

(b) Sunday – the whole weekend, so Friday = 28th.

(c) Saturday – is the weekend beginning 29th in September or not? It starts in September but ends on 1st October. Perhaps the event starts on the 22nd?

(d) Friday – is Friday part of a weekend? Does the event start on the 30th or on the 23rd?

A better specification would be, for example: 'The event takes place on the weekend which includes the last Sunday in September'.

2.

- **First ambiguity:** What does 'next' Wednesday mean, when you are reading on a Monday – Wednesday 6th or Wednesday of next week, the 13th?
- **Second ambiguity:** Does on leave until Wednesday mean that the software engineer's last day of leave is Wednesday, or does it mean that the software engineer will be back on Wednesday?

The colleague might reasonably expect the software engineer next to be back at work on any of: Wednesday 6th, Thursday 7th, Wednesday 13th or Thursday 14th.

3. Some of these questions are answered by user handbooks. Several are not. Answers from a particular user manual:

(a) Invalid dates such as 31st April, day number not accepted.

(b) 29th February – question not dealt with by handbook, but since recorder does not store year it cannot know if 29th February exists or not.

(c) Overlapping requests – not dealt with by handbook.

(d) New Year's Eve – not mentioned, but no problem.

(e) Ordering of requests – not mentioned, but order does not matter.

4. You can't tell, but it matters a lot, because this is genuinely ambiguous. (This was a real specification!)

5. Manuals tend to be unclear; the user may need to experiment to find out what will happen.

Chapter 2

1.

[PERSON] the set of all uniquely identifiable persons

users, loggedIn: ℙPERSON

loggedIn ⊆ users

2.

limit: ℕ

loggedIn ≤ limit

3. Add:

staff, customers: ℙPERSON

and

staff ∩ customers = ∅

staff ∪ customers = users

or

⟨staff, customers⟩ partition users

4.

loggedIn ⊆ staff

#customer > #staff

5.

(a) compulsories ⊆ acceptables

(b) #compulsories = 3

(c) firstAcc ≠ secondAcc

(d) firstAcc ∩ secondAcc ≠ ∅

Chapter 3

1. Invariant property. Only registered users can ever be logged-in.

loggedIn ⊆ users

2. Initialisation operation: no users, no-one logged in. This satisfies the invariant.

users' = ∅
loggedIn' = ∅

3. Add new user. Person *p* must not already be a user. Person *p* is added to *users*.

p: PERSON
p ∉ users
users' = users ∪ {p}
loggedIn' = loggedIn

4. Remove user. Person *p* must already be a user. Person *p* is removed from *users*.

p: PERSON
p ∈ users
p ∉ loggedIn
users' = users \ {p}
loggedIn' = loggedIn

5.

(a) Log in:

p ∈ users
p ∉ loggedIn
loggedIn' = loggedIn ∪ {p}
users' = users

(b) Log out:

p ∈ users
p ∈ loggedIn
loggedIn' = loggedIn \ {p}
users' = users

Chapter 4

1. Law about implication

P	Q	$P \Rightarrow Q$
false	false	true
false	true	true
true	false	false
true	true	true

P	Q	$\neg P$	$\neg P \vee Q$	$P \Rightarrow Q \Leftrightarrow \neg P \vee Q$
false	false	true	true	true
false	true	true	true	true
true	false	false	false	true
true	true	false	true	true

2.

P	Q	$P \Rightarrow Q$	$Q \Rightarrow P$
false	false	true	true
false	true	true	false
true	false	false	true
true	true	true	true

$P \Rightarrow Q \wedge Q \Rightarrow P$	$P \Leftrightarrow Q$
true	true
false	false
false	false
true	true

3.

¬(p ∉ onboard ∧ #onboard < capacity)
⇔
¬(p ∉ onboard) ∨ ¬(#onboard < capacity)
⇔
p ∈ onboard ∨ #onboard ≥ capacity

4.

(a ∧ b) ∨ (a ∧ c) ∨ (a ∧ ¬ c)
⇔ a ∧ (b ∨ c ∨ ¬ c)
⇔ a ∧ (b ∨ true)
⇔ a ∧ true
⇔ a

5. The only way in which

p ∈ loggedIn ∧ p ∈ user

can be true is if both

p ∈ loggedIn

and

p ∈ user

are true. But, because of the given implication, if

p ∈ loggedIn

is true, then so is

p ∈ user

6.

$x \neq 2 \lor x \neq 6$ is
$\neg(x = 2 \land x = 6)$ is
$\neg$false istrue;

any number is either different from 2 or different from 6

7.

$s = t \land s \neq EOF$

if *s* is the same as *t*, and *s* is different from *EOF*, then *t* must be different from *EOF*.

8.

$x \leq y$

9.

$x = 0$

0.

$age < 16 \land \neg student$

Chapter 5

1.

RESPONSE ::=
OK | AlreadyAUser | NotAUser | LoggedIn | NotLoggedIn

2. Add new user:

p: PERSON
reply: RESPONSE

loggedIn' = loggedIn
∧
((p ∉ users ∧
users' = users ∪ {p} ∧
reply = OK)
∨
(p ∈ users ∧
users' = users ∧
reply = AlreadyAUser))

3. Remove user. This answer makes use of the invariant of this system:

loggedIn ⊆ users

which implies that

p ∉ users ⇒ p ∉ loggedIn
p: PERSON
reply: RESPONSE

loggedIn' = loggedIn
∧
((p ∈ users ∧ p ∉ loggedIn ∧
users' = users \ {p} ∧ reply = OK)
∨
(p ∉ users ∧
users' = users ∧ reply = NotAUser)
∨
(p ∈ users ∧ p ∈ loggedIn ∧
users' = users ∧ reply = LoggedIn))

4. Log in:

p: PERSON
reply: RESPONSE

users' = users
∧
((p ∈ users ∧ p ∉ loggedIn ∧
loggedIn' = loggedIn ∪ {p} ∧ reply = OK)
∨
(p ∉ users ∧
loggedIn' = loggedIn ∧ reply = NotAUser)
∨
(p ∈ loggedIn ∧
loggedIn' = loggedIn ∧ reply = LoggedIn))

5. Log out:

p: PERSON
reply: RESPONSE

users' = users
∧
((p ∈ loggedIn ∧
loggedIn' = loggedIn \ {p} ∧ reply = OK)
∨
(p ∉ users ∧
loggedIn' = loggedIn ∧ reply = NotAUser)
∨
(p ∈ users ∧ p ∉ loggedIn ∧
loggedIn' = loggedIn ∧ reply = NotLoggedIn))

Chapter 6

1. *LinesRemaining*

LinesRemaining

ΞCursor
lines!: ℕ

lines! = numLines − line