FACULTY OF
SCIENCE & TECHNOLOGY

COMPUTER SCIENCE AND SOFTWARE ENGINEERING

2013-2014

Code:	ECSE610	**Level:** 6	**Semester:** 2
Title:	Formal Specification		
Date:	20 May 2014		
Time:	10:00		
Duration:	2 Hours		
Module Leader:	Paul Howells		

INSTRUCTIONS TO CANDIDATES

This paper consists of 6 questions.
You must answer ALL questions in Section A and 2 questions from Section B.
Questions in Section A total 50 marks.
All questions in Section B carry 25 marks.
Marks will be awarded to all questions in Section A and the best 2 questions in Section B answered.

Information sheets and additional stationery supplied:

Appendix C contains a summary of the Z notation.

Section A

Answer ALL questions from this section.
You may wish to consult the Z notation given in Appendix C.

Question 1

Given the following Z types and constants declarations:

$$CAR ::= Ford \mid Toyota \mid Vauxhall \mid Honda \mid Renault \mid Ferrari \mid RollsRoyce \mid Bentley$$

$$COUNTRY ::= France \mid Italy \mid Japan \mid UK \mid USA$$

$$\begin{array}{|l} BudgetCars : \mathbb{P}\, CAR \\ LuxuryCars : \mathbb{P}\, CAR \\ \hline BudgetCars = \{\ Ford, Toyota, Vauxhall, Honda, Renault\ \} \\ LuxuryCars = \{\ Ferrari, RollsRoyce, Bentley\ \} \end{array}$$

$$\begin{array}{|l} country : CAR \rightarrow COUNTRY \\ \hline country = \{\ (Ford, USA),\ (Toyota, Japan),\ (Vauxhall, UK), \\ \qquad (Honda, Japan),\ (Renault, France),\ (Ferrari, Italy), \\ \qquad (RollsRoyce, UK),\ (Bentley, UK)\ \} \end{array}$$

Evaluate the following expressions:

(a) $BudgetCars \cup LuxuryCars$ **[1 mark]**

(b) $LuxuryCars \cap \{\ BMW,\ Bentley\ \}$ **[1 mark]**

(c) $\#country$ **[1 mark]**

(d) $country(RollsRoyce)$ **[1 mark]**

(e) $BudgetCars \setminus \{\ Ford, Toyota, Honda, Ferrari\ \}$ **[2 marks]**

(f) dom $country$ **[2 marks]**

(g) ran $country$ **[2 marks]**

(h) $country \rhd \{\ UK\ \}$ **[3 marks]**

(i) $LuxuryCars \lhd country$ **[3 marks]**

(j) $\mathbb{P}\, LuxuryCars$ **[4 marks]**

Question 2

Given the following axiom schema and two state schemas that could be used to specify a family playing the National Lottery.

$$
\begin{array}{|l}
UnLuckyNumber, MaxNumber, MaxLength : \mathbb{N} \\
\hline
UnLuckyNumber = 13 \\
MaxNumber = 49 \\
MaxLength = 6 \\
\end{array}
$$

LuckNumbers

$$
\begin{array}{|l}
Mums, Dads : \mathbb{P}\,\mathbb{N} \\
\hline
UnLuckyNumber \notin Mums \\
MaxNumber \in Dads \\
Mums \cap Dads = \varnothing \\
\end{array}
$$

WinningSequences

$$
\begin{array}{|l}
lastWeeks, thisWeeks : \mathrm{seq}\,\mathbb{N} \\
lastWeeksJackpot, thisWeeksJackpot : \mathbb{N} \\
\hline
lastWeeksJackpot \leq thisWeeksJackpot \\
\#lastWeeks = MaxLength \\
\#thisWeeks = MaxLength \\
\end{array}
$$

(a) Explain when the two state schema conventions Δ and Ξ should be used in a Z specification. **[5 marks]**

(b) Give the expanded versions of the following two schemas:

(i) $\Delta LuckNumbers$ **[4 marks]**

(ii) $\Xi WinningSequences$ **[6 marks]**

Question 3

Given the following Z declarations of the type $LETTER$ and the relations R_1, R_2 & R_3:

$$LETTER ::= a \mid b \mid c \mid d \mid e \mid f \mid g \mid h \mid i \mid j \mid k \mid l \mid m \mid n \mid o \mid p \mid q \mid r \mid s \mid t \mid u \mid v \mid w \mid x \mid y \mid z$$

$$R_1, R_2 : LETTER \leftrightarrow \mathbb{N}$$
$$R_3 : \mathbb{N} \leftrightarrow LETTER$$

$$R_1 = \{ (a,\ 1),\ (b,\ 1),\ (c,\ 3),\ (d,\ 2),\ (e,\ 4),\ (f,\ 4),\ (g,\ 5),\ (h,\ 6) \}$$
$$R_2 = \{ (a,\ 1),\ (b,\ 1),\ (b,\ 2),\ (c,\ 3),\ (d,\ 2) \}$$
$$R_3 = \{ (1,\ x),\ (2,\ y),\ (4,\ z) \}$$

(a) Evaluate the following expressions:

(i) $R_1 \ (\!|\ \{a, c, e\}\ |\!)$ **[2 marks]**

(ii) $R_3 \oplus \{ (0, w),\ (4, a) \}$ **[3 marks]**

(iii) $R_2 \fatsemi R_3$ **[4 marks]**

(b) For each of the relations R_1, R_2 and R_3 state whether it is a just a *relation* or is also a *function*. In addition, if you decide that one of these relations is a function then indicate what kind of function it is, e.g. partial, total, injective, etc. **[6 marks]**

Section B

Answer TWO questions from this section.
You may wish to consult the Z notation given in Appendix C.

Question 4

The following is part of a Library specification.
The following definitions represent the set of books, copies (i.e. instances) of books and borrowers.

$$[\, BOOK,\ COPY,\ BORROWER \,]$$

$$maxloans : \mathbb{N}$$

LibraryDataBase

$$stock : COPY \pfun BOOK$$
$$registeredborrowers : \mathbb{F}\, BORROWER$$

LibraryLoans

$$onloan : COPY \pfun BORROWER$$
$$inlibrary : \mathbb{F}\, COPY$$

$$\forall\, b : BORROWER \bullet \#(onloan \rhd \{\ b\ \}) \leq maxloans$$
$$inlibrary \cap \mathrm{dom}\ onloan = \varnothing$$

Library

$$LibraryDataBase$$
$$LibraryLoans$$

$$\mathrm{dom}\ stock = inlibrary \cup \mathrm{dom}\ onloan$$
$$\mathrm{ran}\ onloan \subseteq registeredborrowers$$

[Continued Overleaf]

IssueBook

$$\Delta Library$$
$$c? : COPY$$
$$b? : BORROWER$$

$$c? \in inlibrary$$
$$stock' = stock$$
$$inlibrary' = inlibrary \setminus \{ c? \}$$
$$\#(onloan \rhd \{ b? \}) < maxloans$$
$$registeredborrowers' = registeredborrowers$$
$$b? \in registeredborrowers$$
$$onloan' = onloan \oplus \{ c? \mapsto b? \}$$

(a) Explain in "plain English" (i.e. do not give a literal translation) the meaning of each line of the following schemas:

(i) *LibraryLoans* **[3 marks]**

(ii) *Library* **[3 marks]**

(b) Explain in "plain English" the meaning of each line of the constraint part of the *IssueBook* schema and the role it plays in the specification of the operation. **[7 marks]**

(c) Specify the *ReturnBook* operation which is used when a borrower returns a book to the library. The specification of this operation must be total and output appropriate success and error reports. In addition the specification should be as modular as possible and make full use of the schema calculus. **[12 marks]**

Question 5

Write a Z specification for a *queue* of people waiting to be "served". For example, at a bank, checkout, etc. Due to the lack of available space the queue has a maximum permitted length.
Your specification should deal with error handling where required and should include the following:

(a) Any types, states and invariants that the queue requires. **[6 marks]**

(b) The queueing operations:

(i) $Join$ – a new person joins the end of the *queue*. **[8 marks]**

(ii) $GetServed$ – the next person gets "served", i.e., leaves the front of the *queue*. **[7 marks]**

(iii) $QueueStatus$ – reports via a suitable message whether the *queue* is *empty*, *full* or *neither full or empty*. **[4 marks]**

Question 6

Part of a Z specification for a University's Computing degree course in particular module registration and de-registration is given in Appendix A.

(a) The ZTC type checker output for the Module specification is given in Appendix B. For each error reported give an explanation and the necessary corrections. **[10 marks]**

(b) Once all of the errors detailed in part **(a)** have been eliminated from the Module specification, explain what additions and modifications must be made to the specification to permit it to be animated by the ZANS animator. **[10 marks]**

(c) Assuming all of the necessary modifications required in part **(b)** have been made, an attempt is made to animate the Module specification in ZANS.

However, ZANS indicates that it can not animate the two operations `RegisterForModule_Ok` and `DeregisterFromModule_Ok`, by reporting that they are *"not explicit"*.

Give a brief explanation of what *"not explicit"* means in this context and state what modifications need to be made to these two operation schemas to correct this problem. **[5 marks]**

Appendix A. Module Specification

A.1 ZTC Box Style Version

```
specification

[ STUDENT ]

TITLE   ::= Formal_Methods | Compilers | Networks

STAFF   ::= P_Howells | A_Lecturer | M_Mouse

REPORT  ::= Success
          | ERROR_Already_Registered
          | ERROR_Module_Full
          | ERROR_Not_Registered

|  MODULE_LIMIT : N

|  ACADEMIC_STAFF : P STAFF

--- Module ---------------------------------
|  title : TITLE ;
|  leader : STAFF ;
|  numbstudents : N ;
|  students : P STUDENT
|--------------------------------
|  leader in ACADEMIC_STAFF ;
|  numbstudents <= MODULE_LIMIT ;
|  numbstudents = students
-----------------------------------------

--- InitialModule ------------------------
|  Module
|--------------------------------
|  title = Formal_Methods ;
|  leader = P_Howells ;
|  numbstudents = 0 ;
|  students = {}
-----------------------------------------

```

```
38          ReportSuccess =^= [ report! : REPORT | report! = Success ]
39
40          --- RegisterForModule_Ok --------------
41          |  Delta Module ;
42          |  newstudent? : STUDENT
43          |-------------------------------
44          |  newstudent? notin students ;
45          |  numbstudents < MODULE_LIMIT ;
46          |  students' = students || newstudent? ;
47          |  numbstudents' = numbstudents + 1 ;
48          ----------------------------------------
49
50          RegisterForModule_Success =^= RegisterForModule_Ok
51                                          /\ ReportSuccess
52
53          --- AlreadyRegistered_Error -----------
54          |  Xi Module ;
55          |  newstudent? : STUDENT ;
56          |  report! : REPORT
57          |-------------------------------
58          |  newstudent? in students
59          |  report! = ERROR_Already_Registered
60          ----------------------------------------------
61
62          --- ModuleFull_Error ----------------------
63          |  Xi Module ;
64          |  newstudent? : STUDENT ;
65          |  report! : REPORT
66          |-------------------------------
67          |  numbstudents = MODULE_LIMIT ;
68          |  report! = ERROR_Module_Full
69          ----------------------------------------
70
71          RegisterForModule  =^= RegisterForModule_Success
72                                  \/ AlreadyRegistered_Error
73                                  \/ ModuleFull_ERROR
74
```

```
75          --- DeregisterFromModule_Ok ---------
76          |  Module ;
77          |  deregstudent? : STUDENT
78          |-------------------------------
79          |  deregstudent? in students ;
80          |  students' = students \ { deregstudent? } ;
81          |  numbstudents' = numbstudents - 1 ;
82          ----------------------------------------
83
84          DeregisterFromModule_Success
85                  =^= DeregisterFromModule_Ok /\ ReportSuccess
86
87          --- Student_Not_Registered_Error ----------
88          |  Xi Module ;
89          |  deregstudent? : STUDENT ;
90          |  report! : REPORT
91          |-------------------------------
92          |  deregstudent? = students ;
93          |  report! = ERROR_Not_Registered
94          ----------------------------------------
95
96          DeregisterFromModule
97                  =^= DeregisterFromModule_Success \/
98                          Student_Not_Registered_Error
99
100         --- PlacesLeftOnModule_Ok --------------------
101         |  Xi Module ;
102         |  placesleft! : N
103         |-------------------------------
104         |  placesleft = MODULE_LIMIT - numbstudents
105         ----------------------------------------------
106
107         PlacesLeftOnModule =^= PlacesLeftOnModule_Ok /\
108                                  ReportSuccess
```

A.2 Pretty-Printed Version

$[STUDENT]$

$TITLE ::= Formal\_Methods \mid Compilers \mid Networks$

$STAFF ::= P\_Howells \mid A\_Lecturer \mid M\_Mouse$

$$REPORT ::= Success \mid ERROR\_Already\_Registered \mid ERROR\_Module\_Full \mid ERROR\_Not\_Registered$$

$$\mid MODULE\_LIMIT : \mathbb{N}$$

$$\mid ACADEMIC\_STAFF : \mathbb{P}\, STAFF$$

Module

$$title : TITLE$$
$$leader : STAFF$$
$$numbstudents : \mathbb{N}$$
$$students : \mathbb{P}\, STUDENT$$

$$leader \in ACADEMIC\_STAFF$$
$$numbstudents \leq MODULE\_LIMIT$$
$$numbstudents = students$$

InitialModule

$$Module$$

$$title = Formal\_Methods$$
$$leader = P\_Howells$$
$$numbstudents = 0$$
$$students = \varnothing$$

$$ReportSuccess \mathrel{\widehat=} [report! : REPORT \mid report! = Success]$$

RegisterForModule_Ok

$\Delta Module$

$newstudent? : STUDENT$

$newstudent? \notin students$

$numbstudents < MODULE\_LIMIT$

$students' = students \cup newstudent?$

$numbstudents' = numbstudents + 1$

$RegisterForModule\_Success \mathrel{\widehat=} RegisterForModule\_Ok \land ReportSuccess$

AlreadyRegistered_Error

$\Xi Module$

$newstudent? : STUDENT$

$report! : REPORT$

$newstudent? \in students$

$report! = ERROR\_Already\_Registered$

ModuleFull_Error

$\Xi Module$

$newstudent? : STUDENT$

$report! : REPORT$

$numbstudents = MODULE\_LIMIT$

$report! = ERROR\_Module\_Full$

$$RegisterForModule \mathrel{\widehat=} RegisterForModule\_Success \lor AlreadyRegistered\_Error \lor ModuleFull\_ERROR$$

DeregisterFromModule_Ok

$Module$

$deregstudent? : STUDENT$

$deregstudent? \in students$

$students' = students \setminus \{deregstudent?\}$

$numbstudents' = numbstudents - 1$

$$DeregisterFromModule\_Success \mathrel{\widehat=} DeregisterFromModule\_Ok \\ \land ReportSuccess$$

Student_Not_Registered_Error

$$\Xi Module$$
$$deregstudent? : STUDENT$$
$$report! : REPORT$$

$$deregstudent? = students$$
$$report! = ERROR\_Not\_Registered$$

$$DeregisterFromModule \mathrel{\widehat=} DeregisterFromModule\_Success \\ \lor Student\_Not\_Registered\_Error$$

PlacesLeftOnModule_Ok

$$\Xi Module$$
$$placesleft! : \mathbb{N}$$

$$placesleft = MODULE\_LIMIT - numbstudents$$

$$PlacesLeftOnModule \mathrel{\widehat=} PlacesLeftOnModule\_Ok \land ReportSuccess$$

Appendix B. ZTC output for Module Specification

The following is part of the ZTC type checker output from the **Module** specification.

```
Parsing main file: module.zbx
... Type checking Given set. "module.zbx" Line 3
... Type checking Free type definition: TITLE. "module.zbx" Line 5
... Type checking Free type definition: STAFF. "module.zbx" Line 7
... Type checking Free type definition: REPORT. "module.zbx" Lines 9-12
... Type checking Axiom box. "module.zbx" Line 14
... Type checking Axiom box. "module.zbx" Line 16
... Type checking Schema box: Module. "module.zbx" Lines 18-26
--- Typing error. "module.zbx" Line 26. Type mismatch:
... Type checking Schema box: InitialModule. "module.zbx" Lines 29-35
... Type checking Schema definition: ReportSuccess. "module.zbx" Line 38
... Type checking Schema box: RegisterForModule_Ok. "module.zbx" Lines 40-47
--- Typing error. "module.zbx" Line 46. Type mismatch: Infix expression
--- Typing error. "module.zbx" Line 46. Type mismatch: Right-hand side
... Type checking Schema definition: RegisterForModule_Success. "module.zbx"
                                     Lines 50-51
... Type checking Schema box: AlreadyRegistered_Error. "module.zbx" Lines 53-59
--- Typing error. "module.zbx" Line 58. Mapping expected:
... Type checking Schema box: ModuleFull_Error. "module.zbx" Lines 62-68
--- Syntax error. "module.zbx" Line 73, near "ModuleFull_ERROR"
... Type checking Schema box: DeregisterFromModule_Ok. "module.zbx" Lines 75-81
--- Typing error. "module.zbx" Line 80. Undefined name: students'
--- Typing error. "module.zbx" Line 81. Undefined name: numbstudents'
... Type checking Schema definition:DeregisterFromModule_Success. "module.zbx"
                                    Lines 84-85
... Type checking Schema box: Student_Not_Registered_Error. "module.zbx" Lines 87-93
--- Typing error. "module.zbx" Line 92. Type mismatch:
... Type checking Schema definition: DeregisterFromModule. "module.zbx" Lines 96-98
... Type checking Schema box: PlacesLeftOnModule_Ok. "module.zbx" Lines 100-104
--- Typing error. "module.zbx" Line 104. Undefined name: placesleft
... Type checking Schema definition: PlacesLeftOnModule. "module.zbx" Lines 107-108
--- Reached the end of the main file while parsing.
End of main file: module.zbx
```

Appendix C. Table of Z Syntax

This appendix contains the Z notation for: sets, logic, ordered pairs, relations, functions, sequences, schemas and the schema calculus.

C.1 Sets

Z Notation	ZTC	Description
$\mathbb{N}$	`N`	Set of natural numbers from 0
$\mathbb{N}_1$	`N1`	Set of natural numbers from 1
$\mathbb{Z}$	`Z`	Set of integers
$x \in S$	`x in S`	x is an element of S
$x \notin S$	`x notin S`	x is not an element of S
$S \subseteq T$	`S subset T`	S is a subset of T
$S \subset T$	`S subseteq T`	S is a strict subset of T
$\varnothing$, $\{\,\}$	`{}`	Empty set
$\mathbb{P}\, S$	`P S`	Power set of S
$\mathbb{F}\, S$	`F S`	Finite power set of S
$S \cup T$	`S || T`	Union of S and T
$S \cap T$	`S && T`	Intersection of S and T
$S \setminus T$	`S \ T`	Set difference of S and T
$\#S$	`#S`	Number of elements in set S
$\{\, D \mid P \bullet E \,\}$	`{ D | P @ E }`	Set comprehension
$\bigcup SS$	`Union SS`	Distributed union of SS
$\bigcap SS$	`Intersection SS`	Distributed intersection of SS
$i \,..\, j$	`i..j`	Range of integers from i to j inclusive
disjoint $\langle A, B, C \rangle$	`disjoint <<A, B, C>>`	Disjoint sets A, B and C
$\langle A, B, C \rangle$ partition S	`<<A, B, C>> partition S`	Sets A, B and C partition the set S

C.2 Logic

Z Notation	ZTC	Description
$\neg P$	`not P`	not P
$P \land Q$	`P and Q`	P and Q
$P \lor Q$	`P or Q`	P or Q
$P \Rightarrow Q$	`P => Q`	P implies Q
$P \Leftrightarrow Q$	`P <=> Q`	P is equivalent to Q
$\forall x : T \bullet P$	`forall x : T @ P`	All elements x of type T satisfy P
$\exists x : T \bullet P$	`exists x : T @ P`	There exists an element x of type T which satisfies P
$\exists_1 x : T \bullet P$	`exists1 x : T @ P`	There exists a *unique* element x of type T which satisfies P

C.3 Ordered Pairs

Z Notation	ZTC	Description
$X \times Y$	`X & Y`	Cartesian product of X and Y
(x, y)	`(x, y)`	Ordered pair
$x \mapsto y$	`x -> y`	Ordered pair, (maplet)
$first(x, y)$	`first(x, y)`	Ordered pair projection function
$second(x, y)$	`second(x, y)`	Ordered pair projection function

C.4 Relations

Z Notation	ZTC	Description
$\mathbb{P}(X \times Y)$	`P(X & Y)`	Set of relations between X and Y
$X \leftrightarrow Y$	`X <-> Y`	Set of relations between X and Y
dom R	`dom R`	Domain of relation R
ran R	`ran R`	Range of relation R
$S \lhd R$	`S <| R`	Domain restriction of R to the set S
$S \lhd R$	`S <+ R`	Domain anti-restriction of R by the set S
$R \rhd S$	`R |> S`	Range restriction of R to the set S
$R \rhd S$	`R +> S`	Range anti-restriction of R by the set S
$R_1 \oplus R_2$	`R1 += R2`	R_1 overridden by relation R_2
$R \fatsemi Q$	`R :> Q`	Relational composition
$R(\!\mid S \mid\!)$	`R (| S |)`	Relational Image of the set S of relation R
id X	`id X`	Identity relation
R^{-1}	`R~`	Inverse relation
R^+	`R^+`	Transitive closure of R
R^*	`R^*`	Reflexive-transitive closure of R

C.5 Functions

Z Notation	ZTC	Description
$\pfun$	++>	Finite function
$\fpfun$	>++>	Finite injection
$\pfun$	+->	Partial function
$\fun$	-->	Total function
$\pinj$	>+>	Partial injection
$\inj$	>->	Total injection
$\psurj$	+>>	Partial surjection
$\surj$	->>	Total surjection
$\bij$	>->>	Bijection

C.6 Sequences

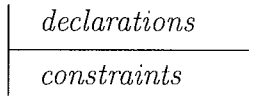
Z Notation	ZTC	Description
$\seq X$	`seq X`	Finite sequences of type X
$\seq_1 X$	`seq1 X`	Non-empty finite sequences of type X
iseq X	`iseq X`	Injective finite sequences of type X
$\langle\rangle$	`<<>>`	Empty sequence
$s \frown t$	`s ^ t`	Concatenation of the sequences s and t
head s	`head s`	First element of a non empty sequence
tail s	`tail s`	All but first element of a non empty sequence
last s	`last s`	Last element of a non empty sequence
front s	`front s`	All but last element of a non empty sequence
rev s	`rev s`	Sequence Reversal
squash s	`squash s`	Sequence Compaction
s prefix t	`s prefix t`	s is a *prefix* of t
s suffix t	`s suffix t`	s is a *suffix* of t
s in t	`s subseq t`	s is a *sub-sequence* of t

C.7 Schema Calculus

Z Notation	ZTC	Description
$[S; D \mid C]$	`[ S; D | C ]`	Schema inclusion
S'	`S'`	Schema decoration
ΔS	`Delta S`	Δ (Delta) Convention
ΞS	`Xi S`	Ξ (Xi) Convention
$S \land T$	`S and T`	Schema Conjunction (S and T)
$S \lor T$	`S or T`	Schema Disjunction (S or T)

C.8 Schemas Types: Z & ZTC Schema Boxes

Axiom Schema



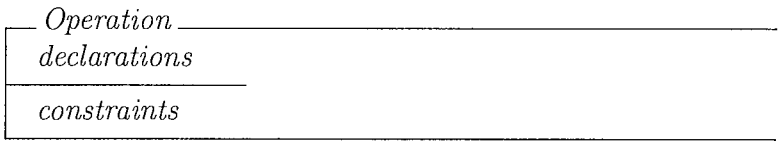

```
| declarations
|--------------
| constraints
```

Linear Schema

$S \mathrel{\widehat=} [declarations \mid constraints]$

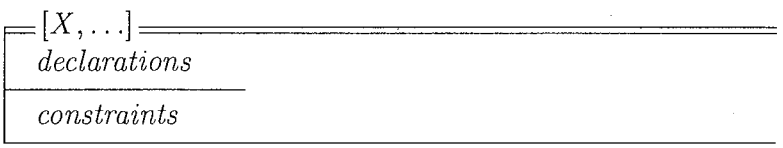
```
S =^= [ declarations | constraints ]
```

State & Operation Schema




```
--- Operation ------------
| declarations
|--------------
| constraints
--------------------------
```

Generic




```
===[X,...]=================
| declarations
|--------------
| constraints
--------------------------
```