UNIVERSITY OF WESTMINSTER

ELECTRONICS AND COMPUTER SCIENCE

2011-2012

Code: ECSE610

Title: Formal Specification

Date: 18 May 2012

Time: 10:00

Duration: 2 Hours

INSTRUCTIONS TO CANDIDATES

Answer ALL questions in Section A and TWO questions from Section B.

Section A is worth a total of 50 marks.

Each question in section B is worth 25 marks.

You may wish to consult the Z notation given in Appendix C.

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 declarations:

$$BIRD ::= Robin \mid Blackbird \mid Magpie \mid Seagull \mid Eagle \mid Penguin \mid Ostrich$$

$$GardenBirds : \mathbb{P}\, BIRDS$$
$$GardenBirds = \{\, Robin, Blackbird, Magpie \,\}$$

$$FlightlessBirds : \mathbb{P}\, BIRDS$$
$$FlightlessBirds = \{\, Penguin, Ostrich \,\}$$

$$sizeOrder : BIRD \rightarrowtail \mathbb{N}$$
$$sizeOrder = \{\, (Robin, 1),\ (Blackbird, 2),\ (Magpie, 3),\ (Seagull, 4),\ (Eagle, 5),\ (Penguin, 6),\ (Ostrich, 7) \,\}$$

Evaluate the following expressions:

- $GardenBirds \cup FlightlessBirds$
- $GardenBirds \setminus \{Magpie\}$
- $\#sizeOrder$
- dom $sizeOrder$
- ran $sizeOrder$
- $sizeOrder(Eagle)$
- $\mathbb{P}\, FlightlessBirds$
- $GardenBirds \lhd sizeOrder$

[10 marks]

Question 2

The following Z declarations and state schemas are part of a University's student and module record system:

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

$$MODULECODE ::= ECSE603 \mid ECSE608 \mid ECSE609 \mid ECSE610$$

$$AllowedModuleLimits : \mathbb{P}\,\mathbb{N}$$

$$AllowedModuleLimits = \{\, 30, 40, 50, 100, 150, 200 \,\}$$

ModuleDetails

$$moduleCode : MODULECODE$$
$$numberOnModule : \mathbb{N}$$
$$moduleLimit : \mathbb{N}$$
$$numberOnModule \leq moduleLimit$$
$$moduleLimit \in AllowedModuleLimits$$

StudentDetails

$$studentLogin : \text{seq}\, LETTER$$
$$studentPassword : \text{seq}\, LETTER$$
$$\langle\,\rangle \neq studentLogin$$
$$6 < \#studentPassword$$

(a) State and explain the formal definitions of ΔS and ΞS for any schema S. **[7 marks]**

(b) Give the *expanded* version of $\Delta ModuleDetails$. **[6 marks]**

(c) Give the *expanded* version of $\Xi StudentDetails$. **[7 marks]**

(d) Define an *initial state schema* for the *StudentDetails* state schema, for a student with the login name *jones* and a suitable password.
(Note: do not use your own password!) **[5 marks]**

Question 3

The Z specification language is often used in the development of "safety critical" software systems. Z is suitable for this task because it allows a system designer to specify important aspects of a system. Within this context of using Z to specify a system, answer the following questions.

(a) There are three *categories* of states that a system can be in, what are they? Illustrate your answer by means of a diagram. **[4 marks]**

(b) What is the *state schema* and what are its two main components? **[2 marks]**

(c) What are *state invariants* and what is their relationship to the three categories of system states? **[2 marks]**

(d) What is the purpose of the *initial state schema* and how does it relate to the *state schema*? **[2 marks]**

(e) When specifying a system operation it is usually necessary to define *pre-conditions* for the operation. What are *pre-conditions* and what is their purpose? **[3 marks]**

(f) The final stage in specifying a system operation is to define the *"total"* version of the operation. Explain what this means. **[2 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 the specification of the *LikeFilms* online film DVD rental system.

$$[\, FILM,\ DVD,\ CUSTOMER \,]$$
$$REPORT ::= Okay$$

$$maxrentals : \mathbb{N}$$

FilmCustomerDataBase

$$stock : DVD \pfun FILM$$
$$customers : \mathbb{P}\, CUSTOMER$$

FilmRentals

$$rentedto : DVD \pfun CUSTOMER$$
$$inwarehouse : \mathbb{P}\, DVD$$

$$\forall\, c : CUSTOMER \bullet \#(rentedto \rhd \{\ c\ \}) \leq maxrentals$$
$$inwarehouse \cap \mathrm{dom}\ rentedto = \varnothing$$

FilmRentalSystem

$$FilmCustomerDataBase$$
$$FilmRentals$$

$$\mathrm{dom}\ stock = inwarehouse \cup \mathrm{dom}\ rentedto$$
$$\mathrm{ran}\ rentedto \subseteq customers$$

$$
\begin{array}{|l}
\textit{RentFilm\_Success} \\
\hline
stock,\ stock' : DVD \pfun FILM \\
customers,\ customers' : \mathbb{P}\, CUSTOMER \\
rentedto,\ rentedto' : DVD \pfun CUSTOMER \\
inwarehouse,\ inwarehouse' : \mathbb{P}\, DVD \\
dvd? : DVD \\
customer? : CUSTOMER \\
report! : REPORT \\
\hline
\mathrm{dom}\ stock = inwarehouse \cup \mathrm{dom}\ rentedto \\
\mathrm{dom}\ stock' = inwarehouse' \cup \mathrm{dom}\ rentedto' \\
\mathrm{ran}\ rentedto \subseteq customers \\
\mathrm{ran}\ rentedto' \subseteq customers' \\
inwarehouse \cap \mathrm{dom}\ rentedto = \varnothing \\
inwarehouse' \cap \mathrm{dom}\ rentedto' = \varnothing \\
dvd? \in inwarehouse \\
customer? \in customers \\
\#(rentedto \rhd \{\ customer?\ \}) < maxrentals \\
stock' = stock \\
customers' = customers \\
rentedto' = rentedto \oplus \{\ dvd? \mapsto customer?\ \} \\
inwarehouse' = inwarehouse \setminus \{\ dvd?\ \} \\
report! = Okay \\
\hline
\end{array}
$$

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

(i) The *invariants* of the $FilmRentalSystem$. **[4 marks]**

(ii) The *pre-conditions* of the $RentFilm\_Success$ operation. **[3 marks]**

(b) Re-specify the $RentFilm\_Success$ operation so that it has a better structure and hence is more readable. This should be achieved by using the Schema Calculus and any new schemas that are required. **[8 marks]**

(c) Based on the version of $RentFilm\_Success$ specified as your answer to part **(b)**:

(i) Explain how the pre-conditions for the error cases of the operation are derived and then derive them. **[3 marks]**

(ii) Define a *"total"* $RentFilm$ operation using the schema calculus and any necessary additional types and schemas. **[7 marks]**

Question 5

Figure 1 represents the design for a web page for an online shopping web site. The four areas correspond to how the web page is to be divided up to represent the four display panels (Sections, Title, Goods and Shopping Cart) for one of the pages for the web site.
The web page window is designed to be 50 columns wide and 20 rows high.

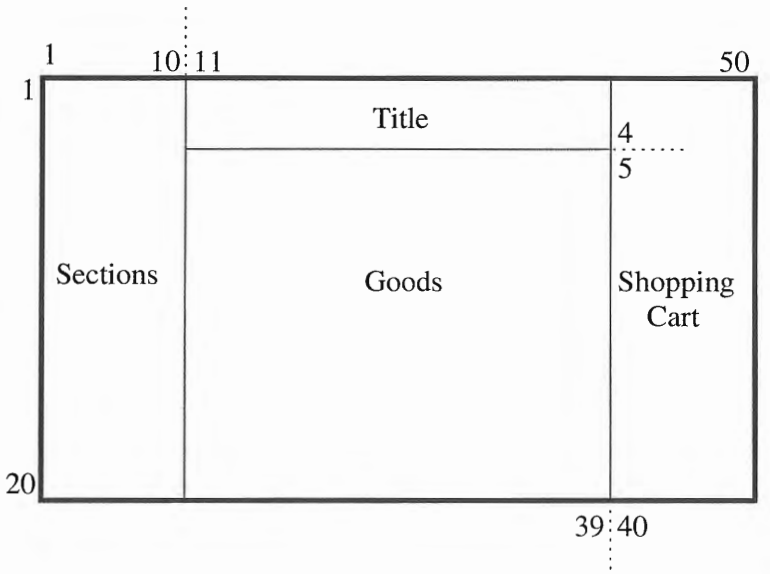


Figure 1: Online Shop's web page display panels

(a) Define suitable Z types and constants to represent the web page window grid as a whole. **[7 marks]**

(b) Define suitable Z constants to represent the four display panel areas within the web page window, i.e., Sections, Title, Goods and Shopping Cart. **[12 marks]**

(c) Specify a *MouseLocationTest* enquiry operation.

This *MouseLocationTest* enquiry operation is to be used to test the system. It simply inputs the location of the mouse within the web page window and outputs the identity of the panel that the mouse is located over. For example, if the mouse location is $(45, 2)$ then it outputs a message to say it is in the *Shopping Cart* panel.

Note that this operation does not have any state, but simply uses the display panel areas to determine the location of the mouse. **[6 marks]**

Question 6

The Z specification of a simple system which records people's birthdays, the *Birthday Book* system, is given in Appendix A.

(a) What are the advantages and disadvantages of using software tools to develop a Z specification? You may support your arguments by referring to any Z tools you have used. **[6 marks]**

(b) The ZTC type checker output for the *Birthday Book* specification is given in Appendix B. For each error give an explanation and the necessary corrections. **[7 marks]**

(c) Assume that all of the errors detailed in part **(b)** have been eliminated from the *Birthday Book* specification. If the specification is to be animated by the ZANS animator it needs to be modified. So with this in mind:

(i) State what ZANS *pragmas* are required and what their purpose is. **[2 marks]**

(ii) Explain what modifications are required to the initial state schema `InitBirthdayBook`. **[6 marks]**

(d) After a Z specification has been loaded into the ZANS animator, and the animation session has been started by means of the `animate` command, ZANS sometimes produces the following message:

```
... Analyzing schema OperationOne: not explicit.
```

where `OperationOne` is an operation schema.

Give a brief explanation of what this message means. In addition describe the features that ZANS provides for finding a solution to this problem. **[4 marks]**

Appendix A. Birthday Book Specification

A.1 Box Style Version

```
specification

[NAME, DATE]

REPORT ::= ok | Already_Known | Not_Known

--- BirthdayBook ----------------------------------------
|  known : P NAME;
|  birthday : NAME +-> DATE
|------------------------
|  known = dom birthday
---------------------------------------------------------

--- InitBirthdayBook ------------------------------------
|  BirthdayBook
|------------------------
|  known = {}
---------------------------------------------------------

--- AddBirthdayOk ---------------------------------------
|  BirthdayBook;
|  name? : NAME;
|  date? : DATE
|------------------------
|  name? notin known;
|  birthday' = birthday || { name? -> date? }
---------------------------------------------------------

ReportSuccess =^= [ result! : REPORT | result! = OK ]

--- AlreadyKnown ----------------------------------------
|  Xi BirthdayBook;
|  name? : NAME;
|  result! : REPORT
|------------------------
|  name? subseteq known;
|  result! = Already_Known
```

```
38          ------------------------------------------------------
39
40          AddBirthday =^= (AddBirthdayOk /\ ReportSuccess) \/ AlreadyKnown
41
42          --- FindBirthdayOk ----------------------------------
43          |  Xi BirthdayBook;
44          |  name? : NAME;
45          |  date! : DATE
46          |------------------------
47          |  name? in known;
48          |  date! = birthday (name?)
49          ------------------------------------------------------
50
51          --- NotKnown ----------------------------------------
52          |  Xi BirthdayBook;
53          |  name? : NAME;
54          |  result! : REPORT
55          |------------------------
56          |  name? notin known
57          |  result! = Not_Known
58          ------------------------------------------------------
59
60          FindBirthday =^= (FindBirthdayOk /\ ReportSuccess) \/ NotKnown
61
62          --- RemindOk ----------------------------------------
63          |  Xi BirthdayBook;
64          |  today? : DATE;
65          |  cards! : NAME
66          |------------------------
67          |  cards! = { n : known | birthday (n) = today? }
68          ------------------------------------------------------
69
70          Remind =^= RemindOk /\ Success
71
72          end specification
```

A.2 Pretty-Printed Version

$[NAME, DATE]$

$REPORT ::= ok \mid Already\_Known \mid Not\_Known$

BirthdayBook

$$known : \mathbb{P}\, NAME$$
$$birthday : NAME \pfun DATE$$
$$known = \mathsf{dom}\, birthday$$

InitBirthdayBook

$$BirthdayBook$$
$$known = \varnothing$$

AddBirthdayOk

$$BirthdayBook$$
$$name? : NAME$$
$$date? : DATE$$
$$name? \notin known$$
$$birthday' = birthday \cup \{name? \mapsto date?\}$$

$ReportSuccess \mathrel{\widehat=} [result! : REPORT \mid result! = OK]$

AlreadyKnown

$$\Xi BirthdayBook$$
$$name? : NAME$$
$$result! : REPORT$$
$$name? \subseteq known$$
$$result! = Already\_Known$$

$AddBirthday \mathrel{\widehat=} (AddBirthdayOk \land ReportSuccess) \lor AlreadyKnown$

FindBirthdayOk

$$\Xi BirthdayBook$$
$$name? : NAME$$
$$date! : DATE$$
$$name? \in known$$
$$date! = birthday(name?)$$

NotKnown

$$\Xi BirthdayBook$$
$$name? : NAME$$
$$result! : REPORT$$
$$name? \notin known$$
$$result! = Not\_Known$$

$$FindBirthday \mathrel{\widehat=} (FindBirthdayOk \land ReportSuccess) \lor NotKnown$$

RemindOk

$$\Xi BirthdayBook$$
$$today? : DATE$$
$$cards! : NAME$$
$$cards! = \{\ n : known \mid birthday(n) = today?\ \}$$

$$Remind \mathrel{\widehat=} RemindOk \land Report$$

Appendix B. ZTC output for Birthday Book

The following is part of the ZTC type check output from the **birthday book** specification.

```
Parsing main file: bdbk.zbx
... Type checking Given Set. "bdbk.zbx" Line 3
... Type checking Free Type Definition: REPORT. "bdbk.zbx" Line 5
... Type checking Schema Box: BirthdayBook. "bdbk.zbx" Lines 7-11
... Type checking Schema Box: InitBirthdayBook. "bdbk.zbx" Lines 14-17
... Type checking Schema Box: AddBirthdayOk. "bdbk.zbx" Lines 20-26
--- Typing error. "bdbk.zbx" Line 26. Undefined name: birthday'
... Type checking Schema Definition: ReportSuccess. "bdbk.zbx" Line 29
--- Typing error. "bdbk.zbx" Line 29. Undefined name: OK
... Type checking Schema Box: AlreadyKnown. "bdbk.zbx" Lines 31-37
--- Typing error. "bdbk.zbx" Line 36. Type mismatch:
>>>name? subseteq known
--- Warning. Indefinite type in schema box.
... Type checking Schema Definition: AddBirthday. "bdbk.zbx" Line 40
... Type checking Schema Box: FindBirthdayOk. "bdbk.zbx" Lines 42-48
... Type checking Schema Box: NotKnown. "bdbk.zbx" Lines 51-57
--- Typing error. "bdbk.zbx" Line 56. Mapping expected:
>>>known
... Type checking Schema Definition: FindBirthday. "bdbk.zbx" Line 60
... Type checking Schema Box: RemindOk. "bdbk.zbx" Lines 62-67
--- Typing error. "bdbk.zbx" Line 67. Type mismatch:
>>>cards! = { n : known | birthday (n) = today? }
--- Syntax error. "bdbk.zbx" Line 70, near "Success"
Expecting: SCHEMANAME PRE NOT '[' '('
>>>
 Remind =^= RemindOk /\ Success
End of main file: bdbk.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
$\twoheadrightarrow$	++>	Finite function
$\rightarrowtail$	>++>	Finite injection
$\rightarrow$	+->	Partial function
$\rightarrow$	-->	Total function
$\rightarrowtail$	>+>	Partial injection
$\rightarrowtail$	>->	Total injection
$\twoheadrightarrow$	+>>	Partial surjection
$\twoheadrightarrow$	->>	Total surjection
$\rightarrowtail\!\!\!\!\twoheadrightarrow$	>->>	Bijection

C.6 Sequences

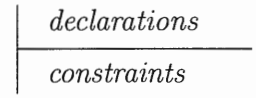
Z Notation	ZTC	Description
$\mathrm{seq}\, X$	`seq X`	Finite sequences of type X
$\mathrm{seq}_1\, X$	`seq1 X`	Non-empty finite sequences of type X
iseq X	`iseq X`	Injective finite sequences of type X
$\langle\,\rangle$	<<>>	Empty sequence
$s \frown t$	`s ^ t`	Concatenation of the sequences s and t
head 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
$S \wedge T$	`S and T`	Schema Conjunction (S and T)
$S \vee T$	`S or T`	Schema Disjunction (S or T)

C.8 Schemas Types: Z & ZTC Schema Boxes

Axiom Schema



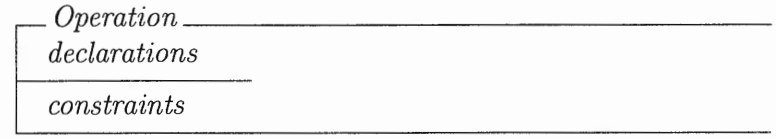

```
| declarations
|--------------
| constraints
```

Linear Schema

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

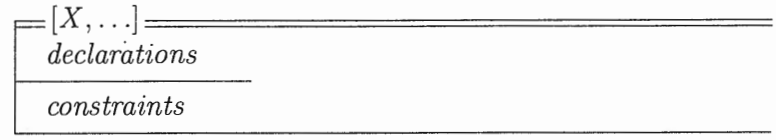
```
S =^= [ declarations | constraints ]
```

State & Operation Schema




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

Generic




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