GX1354

OLLSCOIL NA hÉIREANN
NATIONAL UNIVERSITY OF IRELAND, GALWAY

SEMESTER II, SUMMER 2000 EXAMINATION

Second B.Sc. in Information Technology

Software Engineering I (CT216)

Professor D. Bell
Dr. G. Lyons
Dr. J. Duggan
Dr. M. McGettrick

Time Allowed: 3 hours
Answer any five questions
Use a separate answer book for each section

Section A: Structured Techniques

1. Assume that a system routine, named *GetEnvironmentInfo,* is available, and it returns useful information about an application's environment. This information is returned as a record (labeled EnvInfo), which is defined as:

   EnvInfo = Processor Type + Processor Speed +Memory Size + Network Address

   Assume that your application requires separate access to each of these categories of information (e.g. it may only require the processor type at one part of the code, and the network address at another segment of the code). Furthermore, assume that, due to performance concerns, *GetEnvironmentInfo* should only be called once.

   Design an information cluster as a solution, and specify this using *structure charts* and *routine specifications.*

2. Write notes on any *two* of the following topics:

   - The Software Development Life Cycle
   - Control Coupling
   - Requirements Gathering
   - The Environmental Model

3. Consider the relation below and:

- Identify the primary key
- Establish functional dependencies
- Convert to 3NF

Customer ID	Customer Name	SalesPerson ID	SalesPersonName	Region
CR0001	Food Suppliers Inc.	SP0001	Mary Manning	West
CR0001	Food Suppliers Inc.	SP0002	John O'Neill	West
CR0002	Drink Suppliers Inc.	SP0001	Mary Manning	West
CR0003	Drink Suppliers Inc.	SP0003	Bill Murphy	North
CR0003	Drink Suppliers Inc.	SP0001	Mary Manning	North
CR0003	Drink Suppliers Inc.	SP0002	John O'Neill	North
CR0003	Cutlery Suppliers Inc.	SP0003	Bill Murphy	West

4. For the routine specified below:

- Represent it as a structure chart
- Represent it as a flow graph
- Calculate V(G) using 3 methods

```
//  Routine:      generateLottoNumbers
//  Updates:      a – array of 6 integers
//  Purpose:      generates 6 unique numbers in the range [1-42], stores them
//                in the array a, and returns control to the calling routine

Begin
    int alreadyExists = 0;
    int i = 0, j = 0;
    int numberCount= 0, newNumber = 0;

    while (numberCount < 6)
    {
            // call pre-defined function
            newNumber = getRandomNumber(1,42);

            j = 0; alreadyExists = 0;
            while (j < numberCount)
            {
                    if(a[j] == newNumber)
                            alreadyExists = 1;
            }

            if(alreadyExists == 0)
            {
              a[numberCount] = newNumber;
              numberCount++;
            }
    }
End
```

5. For the *University Information System:*

   - Draw an entity-relationship diagram.
   - Convert the E-R model to relations.

   Lecturers (staff id, name, date of birth) occupy an office (office code, building description and floor number). A single office is assigned to each lecturer. Students (student id, name, date of birth) can have one or more addresses (house number, street, town, county). Students study many subjects (subject code, description and maximum marks), and subjects are taken by more than one student. Information on a students result in each subject must be stored. Lecturers teach subjects in theatres (theatre code, capacity). A subject may be taught by more than one lecturer, and lecturers usually teach more than one subject. The day of the week (Monday-Friday) and the time of the lecture must also be stored.

6. As part of a College Information system, *Generate Result Reports* is a system response to a temporal event *Students Request Exam Results*. Based on the description of this response, produce:

   - A DFD of the response
   - A data dictionary
   - A process specification

   For each student in the system, results are processed and stored in a data store called REPORTS. Each report consists of the students: id, name and address; and each subject code, description and result (in percentages). There are three other stores in the system: STUDENTS, RESULTS and SUBJECTS. Each student is described by a unique student id, a name and address. Every result is uniquely identified by a combination of the student id and the subject code, and the marks awarded. Each subject is uniquely identified by a subject code, and also has a description and maximum marks.

Section B: Formal Methods

7. Given the basic types $[PERSON, PARKING\_SLOT]$ and the state schema

$$
\begin{array}{|l}
\textit{Parking} \\
\hline
staff : \mathbb{P}\, PERSON \\
assigned\_to : PARKING\_SLOT \leftrightarrow PERSON \\
available, occupied : \mathbb{P}\, PARKING\_SLOT \\
\hline
\operatorname{ran} assigned\_to \subseteq staff \\
available \cap occupied = \emptyset \\
\hline
\end{array}
$$

write schema (with appropriate error handling) to

(a) allocate a parking slot to a new member of staff.

(b) de-allocate a parking slot from a staff member who has left the company.

(c) construct a new *Parking* schema which sets an upper limit on the size of the Parking Lot and categorizes staff as either senior (who are assigned 2 parking slots) or junior (who are assigned one).

8. To model a rudimentary text editor we can represent the position of the cursor by specifying a sequence of characters to the left and right. Using this idea or otherwise, write a formal specification in Z (using the given type $[Char]$) with operations to

(a) move to the left.

(b) move to the right.

(c) insert to the right.

(d) delete to the left.

Glossary of Z notation

Names

a,b	identifiers
d,e	declarations (e.g., $a : A$; $b, \ldots : B \ldots$)
f,g	functions
m,n	numbers
p,q	predicates
s,t	sequences
x,y	expressions
A,B	sets
C,D	bags
Q,R	relations
S,T	schemas
X	schema text (e.g., d, $d \mid p$ or S)

Definitions

$a == x$	Abbreviation definition
$a ::= b \mid \ldots$	Free type definition (or $a ::= b\langle\langle x \rangle\rangle \mid \ldots$)
$[a]$	Introduction of a given set (or $[a, \ldots]$)
$a\_$	Prefix operator
$\_a$	Postfix operator
$\_a\_$	Infix operator

Logic

$true$	Logical true constant
$false$	Logical false constant
$\neg\, p$	Logical negation
$p \land q$	Logical conjunction
$p \lor q$	Logical disjunction
$p \Rightarrow q$	Logical implication ($\neg\, p \lor q$)
$p \Leftrightarrow q$	Logical equivalence ($p \Rightarrow q \land q \Rightarrow p$)
$\forall X \bullet q$	Universal quantification
$\exists X \bullet q$	Existential quantification
$\exists_1 X \bullet q$	Unique existential quantification
let $a == x$; ... $\bullet$ p	Local definition

Sets and expressions

$x = y$	Equality of expressions
$x \neq y$	Inequality ($\neg\, (x = y)$)
$x \in A$	Set membership
$x \notin A$	Non-membership ($\neg\, (x \in A)$)
$\varnothing$	Empty set
$A \subseteq B$	Set inclusion
$A \subset B$	Strict set inclusion ($A \subseteq B \land A \neq B$)
$\{x, y, \ldots\}$	Set of elements
$\{X \mid x\}$	Set comprehension
$\lambda X \mid x$	Lambda-expression – function
$\mu X \mid x$	Mu-expression – unique value
let $a == x$; ... $\bullet$ y	Local definition
if p then x else y	Conditional expression
$(x, y, \ldots)$	Ordered tuple
$A \times B \times \ldots$	Cartesian product
$\mathbb{P}\, A$	Power set (set of subsets)
$\mathbb{P}_1\, A$	Non-empty power set
$\mathbb{F}\, A$	Set of finite subsets
$\mathbb{F}_1\, A$	Non-empty set of finite subsets
$A \cap B$	Set intersection
$A \cup B$	Set union
$A \setminus B$	Set difference
$\bigcup A$	Generalized union of a set of sets
$\bigcap A$	Generalized intersection of a set of sets
$first\ x$	First element of an ordered pair
$second\ x$	Second element of an ordered pair
$\#A$	Size of a finite set

Relations

$A \leftrightarrow B$	Relation ($\mathbb{P}(A \times B)$)
$a \mapsto b$	Maplet ((a, b))
dom R	Domain of a relation
ran R	Range of a relation
id A	Identity relation
$Q \fatsemi R$	Forward relational composition
$Q \circ R$	Backward relational composition ($R \fatsemi Q$)
$A \lhd R$	Domain restriction
$A \lhd\!\!\!\!- R$	Domain anti-restriction
$R \rhd A$	Range restriction
$R \rhd\!\!\!\!- A$	Range anti-restriction
$R(\!\mid A \mid\!)$	Relational image
$iter\ n\ R$	Relation composed n times
R^n	Same as $iter\ n\ R$
$R^\sim$	Inverse of relation (R^{-1})
R^*	Reflexive-transitive closure
R^+	Irreflexive-transitive closure
$Q \oplus R$	Relational overriding ($(\text{dom}\, R \lhd\!\!\!\!- Q) \cup R$)
$a\ \underline{R}\ b$	Infix relation

Functions

$A \pfun B$	Partial functions
$A \fun B$	Total functions
$A \pinj B$	Partial injections
$A \inj B$	Total injections
$A \psur B$	Partial surjections
$A \sur B$	Total surjections
$A \bij B$	Bijective functions
$A \ffun B$	Finite partial functions
$A \finj B$	Finite partial injections
$f\ x$	Function application (or $f(x)$)

Numbers

$\mathbb{Z}$	Set of integers
$\mathbb{N}$	Set of natural numbers $\{0, 1, 2, ...\}$
$\mathbb{N}_1$	Set of non-zero natural numbers $(\mathbb{N} \setminus \{0\})$
$m + n$	Addition
$m - n$	Subtraction
$m * n$	Multiplication
$m \operatorname{div} n$	Division
$m \bmod n$	Modulo arithmetic
$m \leq n$	Less than or equal
$m < n$	Less than
$m \geq n$	Greater than or equal
$m > n$	Greater than
$succ\ n$	Successor function $\{0 \mapsto 1, 1 \mapsto 2, ...\}$
$m\ ..\ n$	Number range
$min\ A$	Minimum of a set of numbers
$max\ A$	Maximum of a set of numbers

Sequences

$\operatorname{seq} A$	Set of finite sequences
$\operatorname{seq}_1 A$	Set of non-empty finite sequences
$\operatorname{iseq} A$	Set of finite injective sequences
$\langle\rangle$	Empty sequence
$\langle x, y, ...\rangle$	Sequence $\{1 \mapsto x, 2 \mapsto y, ...\}$
$s \frown t$	Sequence concatenation
$\frown / s$	Distributed sequence concatenation
$head\ s$	First element of sequence ($s(1)$)
$tail\ s$	All but the head element of a sequence
$last\ s$	Last element of sequence ($s(\#s)$)
$front\ s$	All but the last element of a sequence
$rev\ s$	Reverse a sequence
$squash\ f$	Compact a function to a sequence
$A \upharpoonright s$	Sequence extraction ($squash(A \lhd s)$)
$s \upharpoonright A$	Sequence filtering ($squash(s \rhd A)$)
s prefix t	Sequence prefix relation ($s \frown v = t$)
s suffix t	Sequence suffix relation ($u \frown s = t$)
s in t	Sequence segment relation ($u \frown s \frown v = t$)
disjoint A	Disjointness of an indexed family of sets
A partition B	Partition an indexed family of sets

Bags

$\operatorname{bag} A$	Set of bags or multisets ($A \nrightarrow \mathbb{N}_1$)
$[\![]\!]$	Empty bag
$[\![x, y, ...]\!]$	Bag $\{x \mapsto 1, y \mapsto 1, ...\}$
$count\ C\ x$	Multiplicity of an element in a bag
$C \sharp x$	Same as $count\ C\ x$
$n \otimes C$	Bag scaling of multiplicity
$x \sqsubseteq C$	Bag membership
$C \sqsubseteq D$	Sub-bag relation
$C \uplus D$	Bag union
$C \setminus\!\!\setminus D$	Bag difference
$items\ s$	Bag of elements in a sequence

Schema notation

Vertical schema.

S
d
p

New lines denote ';' and '∧'. The schema name and predicate part are optional. The schema may subsequently be referenced by name in the document.

Axiomatic definition.

d
p

The definitions may be non-unique. The predicate part is optional. The definitions apply globally in the document.

$[a, ...]$
d
p

Generic definition.

The generic parameters are optional. The definitions must be unique. The definitions apply globally in the document.

$S \mathrel{\widehat=} [X]$	Horizontal schema
$[T; ... \mid ...]$	Schema inclusion
$z.a$	Component selection (given $z : S$)
θS	Tuple of components
$\neg\ S$	Schema negation
$\operatorname{pre} S$	Schema precondition
$S \wedge T$	Schema conjunction
$S \vee T$	Schema disjunction
$S \Rightarrow T$	Schema implication
$S \Leftrightarrow T$	Schema equivalence
$S \setminus (a, ...)$	Hiding of component(s)
$S \upharpoonright T$	Projection of components
$S \fatsemi T$	Schema composition (S then T)
$S \gg T$	Schema piping (S outputs to T inputs)
$S[a/b, ...]$	Schema component renaming (b becomes a, etc.)
$\forall X \bullet S$	Schema universal quantification
$\exists X \bullet S$	Schema existential quantification
$\exists_1 X \bullet S$	Schema unique existential quantification

Conventions

$a?$	Input to an operation
$a!$	Output from an operation
a	State component before an operation
a'	State component after an operation
S	State schema before an operation
S'	State schema after an operation
ΔS	Change of state (normally $S \wedge S'$)
ΞS	No change of state (normally $[S \wedge S' \mid \theta S = \theta S']$)