PAPER CODE NO.	EXAMINER : **Professor M Fisher**
COMP313	DEPARTMENT : Computer Science Tel. No. **46701/7663**

THE UNIVERSITY
of LIVERPOOL

SUMMER 2002 EXAMINATIONS

Bachelor of Arts : Year 3
Bachelor of Science : Year 3

FORMAL METHODS

TIME ALLOWED : Two Hours and a Half

INSTRUCTIONS TO CANDIDATES

Answer ***four*** questions only

If you attempt to answer more than the required number of questions (in any section), the marks awarded for the excess questions will be discarded (starting with your lowest mark).

THE UNIVERSITY
of LIVERPOOL

1. This question concerns the basic structures used within Z specifications.

   (a) What is the representation of the sequence

   $$\langle 1, 3, 5, 7, 9 \rangle$$

   when given in terms of a function (i.e., as a set of maplets)? **[3]**

   (b) What is the representation of the bag

   $$[\![pig, cow, horse, hen, pig, horse, duck]\!]$$

   when given in terms of a function (i.e., as a set of maplets)? **[3]**

   (c) Given $c : \mathbb{P}\{a, b, d, g\}$, write down all the values c can possibly have. **[4]**

   (d) f is a function from $\{1, 2, 3, 4\}$ to $\{a, b, c\}$ that is defined as:

   $$f == \{1 \mapsto a, 2 \mapsto b, 3 \mapsto a, 4 \mapsto c\}$$

   Is f surjective or injective, and why? **[5]**

   (e) What is the value of

   $$\langle a, b, r, a, c, a, d, a, b, r, a \rangle \upharpoonright (\{a, b, c\} \setminus (\mathrm{dom}\{a \mapsto 2, d \mapsto 4\}))$$

   Explain your answer. **[5]**

   (f) If $B == [\![k, c, a, j, k, c, a, l, b]\!]$, then what is the value of

   $$(B \oplus \{c \mapsto 5\}) \setminus \{j \mapsto 1\}$$

   [5]

2. We wish to specify the relationship between exam marks and students, and already have the following state space schema (N.B., *PERSON* is the set of all people):

$$\begin{array}{|l}
\hline
\textit{MarkRecord} \\
students : \mathbb{P}\, PERSON \\
marks : PERSON \pfun 0 \mathinner{..} 100 \\
\hline
\operatorname{dom} marks \subseteq students \\
\hline
\end{array}$$

(a) Write a Z specification for the operation

$$AddStudent(name? : PERSON)$$

which adds a new student (i.e. *name?*) to *MarkRecord*, but does not assign the student a mark. **[5]**

(b) Write a Z specification for the operation

$$AddMark(name? : PERSON, mark? : 0 \mathinner{..} 100)$$

which assigns the mark (*mark?*) to the student (*name?*) in the *MarkRecord*. **[5]**

(c) Write a Z specification for the operation

$$CheckMark(name? : PERSON, mark! : 0 \mathinner{..} 100)$$

which returns the mark (*mark!*) associated with the student (*name?*).

Note: the operation should be undefined if the given student has not already been assigned a mark. **[5]**

(d) Write a Z specification for the operation

$$Unmarked(names? : \mathbb{P}\, PERSON)$$

which returns the set of students that have not yet been assigned marks. **[5]**

(e) How would you modify the *CheckMark* operation above so that it is robust (i.e. it will be defined for any student name supplied)? Assume that a *REPORT* type exists for reporting errors. **[5]**

THE UNIVERSITY
of LIVERPOOL

3. This question concerns temporal logic.

   (a) What type of structure is typically used to provide a model for propositional, discrete, linear temporal logic, with finite past, and why? **[3]**

   (b) Does the temporal formula

   $$(l \wedge a)\,\mathcal{U}\,(l \wedge\ (l \wedge b)\,\mathcal{U}\,c)$$

   imply $l\,\mathcal{U}\,c$? Explain your answer. **[5]**

   (c) Does the temporal formula

   $$\Box(p \Rightarrow \Diamond q) \wedge \Box p$$

   imply $\Box q$? Explain your answer. **[5]**

   (d) Consider the axiom

   $$(p\ \wedge\ \Box(p \Rightarrow \bigcirc p))\ \Rightarrow\ \Box p$$

   Translate this to classical first-order logic (with arithmetic) and explain what simple principle the above axiom characterises. **[8]**

   (e) How do *branching* temporal logics differ from linear temporal logics, and what additional operators do they typically provide? **[4]**

THE UNIVERSITY
of LIVERPOOL

4. Below is a temporal specification for a simple message-passing system consisting of two components, A and B.

$$Spec_A\colon \Box \left[\begin{array}{lrcl} & \text{start} & \Rightarrow & p \\ \wedge & p & \Rightarrow & \bigcirc q \\ \wedge & q & \Rightarrow & \bigcirc p \\ \wedge & q & \Rightarrow & \bigcirc send\_msg \end{array}\right] \qquad Spec_B\colon \Box \left[\begin{array}{lrcl} & rcv\_msg & \Rightarrow & \bigcirc g \\ \wedge & f & \Rightarrow & \bigcirc g \\ \wedge & g & \Rightarrow & \bigcirc f \end{array}\right]$$

(a) What is the behaviour of $Spec_A$, i.e. how often is $send\_msg$ made true? **[7]**

(b) In

$$Spec_A \wedge Spec_B \wedge \Box[send\_msg \Rightarrow rcv\_msg]$$

what is the last formula meant to specify? **[3]**

(c) If we wish to specify that a message send will be followed, at some time in the future, by a message receipt, what formula should we modify in the above specification and what should it be changed to? **[5]**

(d) What is a *safety* property, and what general form of temporal formulae characterise such properties? **[5]**

(e) What is a *liveness* property, and what general form of temporal formulae characterise such properties? **[5]**

5. This question concerns the foundations of model checking.

(a) Given a finite state structure, M, represented as a finite-state automaton, and a temporal formula, φ, how would we use the *automata-theoretic* approach to model checking to establish $M \models \varphi$? **[10]**

(b) What is *on the fly* model checking, and why might it be beneficial? **[7]**

(c) Describe two current problems with the model checking approach in general. **[8]**

6. Consider the following Promela code describing a three process system where:

   - process A sends information to process B via channel a2b,
   - process B sends information to process C via channel b2c, and
   - process C sends information to process A via channel c2a.

```
proctype A (chan in, out)
{
        int total;
        total = 0;                          /* initial state */
S1:     total = (total+1)%8;
        out!total;
        printf("A sent %d\n", total);
        in?total;
        assert(total != 1);                    /* assertion */
        printf("A received %d\n", total);
        if
        :: (total != 0) -> goto S1;
        :: (total == 0) -> out!total
        fi }

proctype B (chan in, out)
{
        int total;
S1:     in?total;
        printf("B received %d\n", total);
        if
        :: (total != 0) -> total = (total+1)%8;  out!total;
                           printf("B sent %d\n", total);
                           goto S1;
        :: (total == 0) -> out!total
        fi }

proctype C (chan in, out)
{
        int total;
S1:     in?total;
        printf("C received %d\n", total);
        if
        :: (total != 0) -> total = (total+1)%8;  out!total;
                           printf("C sent %d\n", total);
                           goto S1;
        :: (total == 0) -> out!total
        fi }

init {  chan a2b = [1] of { int };
        chan b2c = [1] of { int };
        chan c2a = [1] of { int };
        atomic { run A(c2a, a2b); run B(a2b, b2c); run C(b2c, c2a) }
     }
```

Note that `%` is the modulo arithmetic operator. So, for example `(17%8)=1` and `(15%8)=7`.

(a) If we execute this program what sequence of outputs can we expect? **[8]**

(b) Will the assertion in process `A` succeed? Explain your answer. **[7]**

(c) What will happen if we change the assertion in process `A` to be

```
assert(total != 3)
```

[5]

(d) If we wanted to verify that it is *not* the case that the `total` variable has a non-zero value infinitely often, what temporal formula would we wish to check? **[5]**