NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING
EXAMINATION FOR
Semester 2, 2008/2009

CS 5201 - FOUNDATION IN THEORETICAL CS

April/May 2009 Time Allowed: 3 Hours

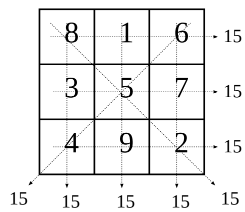
INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **four**(**4**) long questions and comprises **five** (**5**) pages, including this page.
2. *Answer* **three** *out of* **four** *questions.*
3. Each question should be answered in a **separate** answer book.
4. This is an *OPEN BOOK* examination.
5. Write your Matriculation number in **all** the answer books.

A: Algorithm (10 marks)

(A) **Magic Squares**

(3 marks) A magic square of order n is an arrangement of the numbers from 1 to n^2 in an n-by-n matrix, with each number occurring exactly once, so that each row, each column, and each main diagonal has the same sum. The figure below shows an example 3-by-3 magic square. Prove that if a magic square of order n exists, the sum in question must be equal to $(n(n^2+1))/2$.



(B) **Mode of a List of Integers**

(3 marks) A *mode* of a list of integers is an element that occurs at least as often as each of the other elements. Devise an algorithm that finds a mode in a list of n nondecreasing integers. For example, the mode of the list $\{1, 3, 4, 4, 5, 5, 5, 6, 9\}$ is 5. Identify what the complexity of your algorithm is in big oh notation, $O(f(n))$, where $f(n)$ is one of the usual complexity classes (e.g., $\log n$, n, $n \log n$, n^2, $\ldots$).

(C) **Lucas Numbers**

(4 marks) Lucas numbers L_n are a sequence of numbers that are produced by the following definition:

$$\begin{aligned} L_n &= L_{n-1} + L_{n-2} \quad \text{for } n > 1 \\ L_0 &= 2 \\ L_1 &= 1 \end{aligned}$$

Consider the pseudo code algorithms Lucas1(n), Lucas2(n) and Lucas3(n) to compute the Lucas numbers (note, Fibonacci(n) is a helper function for Lucas3(n)). Describe which of the three procedures is the most efficient and which one is the least efficient. Explain your answer.

Algorithm 1 Lucas1(n)

if $n = 0$ **then return** 2;
else if $n = 1$ **then return** 1;
else return Lucas1(n-1) + Lucas1(n-2);

Algorithm 2 Lucas2(n)

L[0] $\leftarrow$ 2; L[1] $\leftarrow$ 1;
for $i \leftarrow 2$ **to** n
    L[i] $\leftarrow$ L[i-1] + L[i-2];
return L[n];

Algorithm 3 Lucas3(n)

if $n = 0$ **then return** 2;
else return Fibonacci(n-1) + Fibonacci(n+1);

Algorithm 4 Fibonacci(n)

if $n \leq 1$ **return** n;
else return Fibonacci(n-1) + Fibonacci(n-2);

B: Theory of Computation (10 marks)

(A) A regular expression describes a regular set of strings; expressions can be formed by listing finite set of strings, taking the star-operation of another regular expression, taking the union of regular expressions and taking the concatenation of regular expressions. Make a nondeterministic finite automaton consisting of up to 5 states accepting the set given by the following regular expression:

$$(\{1,2,3,4,5,6,7,8,9\} \cdot \{0,1,2,3,4,5,6,7,8,9\}^* \cdot \{08,33,58,83\}) \cup \{8,33,58,83\}.$$

(B) Recall that a composite number is a natural number consisting of two non-trivial factors; the smallest composite numbers are $4 = 2 \cdot 2$, $6 = 2 \cdot 3$, $8 = 2 \cdot 4$ and $9 = 3 \cdot 3$. Let L be the set of all words over the alphabet $\{0, 1\}$ whose length is a composite number. Determine the level in the Chomsky hierarchy (r.e., context-sensitive, context-free, regular) which L takes. Prove that L goes exactly onto the level chosen (and not below or above).

(C) Consider the following statement: "There is a language $L \subseteq \{0, 1, 2\}^*$ such that L is accepted by a nondeterministic finite automaton having 1819 states but not by a nondeterministic finite automaton having 1818 states." Write whether the statement is true or false and prove your answer.

C: Principles of Programming Languages (10 marks)

(1). Pure lambda calculus can be constructed using the following grammar rules:

$$\begin{array}{lll} e ::= & x & \textit{variable} \\ & \mid \lambda \cdot e & \textit{function abstraction} \\ & \mid e_1\ e_2 & \textit{application} \end{array}$$

Explain why this calculus is sometimes being referred to as the simplest universal programming language. [2 marks]

(2). The following lambda term is often referred to as a fix-point operator. (3 marks)

$$\text{fix} = \lambda\ \text{f}\ \cdot\ (\lambda\ \text{x}\ \cdot\ \text{f (x x)})\ (\lambda\ \text{x}\ \cdot\ \text{f (x x)})$$

An operator g is said to be a fix-point operator if we can prove the following property:

$$\text{g h} = \text{h}\ \ \text{(g h)}$$

(i) Prove that fix has this property.

(ii) Rewrite the following function to a non-recursive counterpart with the help of the above fix operator.

```
add = λ x · λ y · if x==0 then y
                  else y+(add (x-1) y)
```

(3). Consider a simple language below (3 marks)

$$\begin{array}{ll} e ::= & x \\ & \mid \texttt{Int}\ i \mid \texttt{add}\ e_1\ e_2 \mid \texttt{time}\ e_1\ e_2 \\ & \mid (e_1, e_2) \mid \texttt{fst}\ e \mid \texttt{snd}\ e \\ & \mid \lambda\ x \cdot e \mid e_1\ e_2 \\ & \mid \texttt{letrec}\ x = e_1\ \texttt{in}\ e_2 \end{array}$$

Local variables x may be introduced by lambda abstraction ($\lambda\, x \cdot e$) and a recursive let construct ($\texttt{letrec}\ x = e1\ \texttt{in}\ e2$). These local variables are assumed to be lexically bound, and may shadow previous occurrences of the same local variable.

As an example, the following program fragment has a clash in the local variable v which led to its shadowing.

$$\texttt{letrec}\ v\ =\ (\lambda\ v \cdot v)\ \texttt{in}\ (v\ 3)$$

To avoid such clashes in bound variables, we may uniquely rename the inner occurrence of variable v to:

$$\texttt{letrec}\ v\ =\ (\lambda\ z \cdot z)\ \texttt{in}\ (v\ 3)$$

Define a translation function (over the corresponding abstract syntax tree of the given language) that would detect clashes in local variables, and provide suitable renaming whenever a clash occur. You may assume a function fresh_var that will automatically generate a new variable name.

(4). Describe clearly the key differences between mechanisms of "parametric types" and "overloading". Explain how parametric types could be supported in an object-oriented programming language, such as Java. (2 marks)

D: Logic and AI (60 marks)

Problem 1 (5 + 5 = 10 marks)
Transform the following set of formulas into clausal form and refute using resolution.
$\{p,\ p \rightarrow ((t \vee r) \wedge (\sim q \vee \sim r)),\ t \vee q,\ \sim t\}$.

Problem 2 (10 marks)
Consider an inference rule of first order logic of the form:

$$\frac{\vdash\ A}{\vdash\ B}.$$

Such a rule is said to be *sound* if A is valid implies that B is valid too.

Show that the following inference rule for first order logic is sound.

$$\frac{\vdash A(a) \rightarrow B(a)}{\vdash \forall x A(x) \rightarrow \forall x B(x)}$$

Problem 3 (20 marks)
Let EQ denote the equality predicate of first order logic. In other words for every domain U, the predicate EQ will be interpreted over this domain to be the binary relation $\{(u, u) \mid u \in U\}$.

1. Express in first order logic a sentence describing the fact that EQ is an equivalence relation.
2. Use EQ to form a sentence which is satisfiable in the domain U iff U has exactly 2 elements.

Problem 4 (20 marks)

1. Use the semantic tableau method to show that the following first order sentence is valid. $(\exists x(A(x) \rightarrow B(x))) \rightarrow ((\forall x(A(x)) \rightarrow (\exists x(B(x)))$
2. Show that the following first order sentence is not valid by exhibiting a falsifying model.

   $(\exists x A(x) \rightarrow \exists x B(x)) \rightarrow \forall x(A(x) \rightarrow B(x))$