NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 1, 2006/2007

CS5201 - FOUNDATION IN THEORETICAL CS

November 2006 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. **Remember to write your Matriculation number on each answer book.**
5. This is an **OPEN BOOK** examination.

1. Theory of Computation

A Consider the set defined by the following regular expression:

$$0 \bigcup 1 \cdot (\{0,2\}^* \cdot 1 \cdot \{0,2\}^* \cdot 1)^* \{0,2\}^* \cdot 1 \cdot \{0,2\}^* \bigcup 2 \cdot (\{0,2\}^* \cdot 1 \cdot \{0,2\}^* \cdot 1)^* \{0,2\}^*$$

1. What set of numbers in the ternary system (*i.e.*, number system with base 3) does this expression represent?

B Given is a context-free grammar with nonterminals S, X, Y, start symbol S and rules

$S \to XXXX$

$X \to YYYY$

$Y \to 0Y$

$Y \to Y0$

$Y \to 1$

1. How many 1s and 2s can a word in this language have?

2. Give a regular expression for the language.

3. How many states would a complete and deterministic finite automaton need to accept this language: 8, 18, 28, 38?

C Write context-free grammars accepting the following regular languages:

1. 0^+

2. $0^+1^+2^+0^+1^+2^+0^+1^+2^+$

3. $(0+1+2)^{81}$

D A language L satisfies the Pumping Lemma iff there is a constant c such that for every word $w \in L$ consisting of at least c symbols, one can represent $w = xyz$ with xy having at most c symbols, y having at least one symbol and $x(y^*)z \subseteq L$. Determine the minimal possible c with which the following languages satisfy the Pumping Lemma (no explanation is required).

1. 0^+1^+

2. $(00+11+22)^*$

2. Logic and Artificial Intelligence

A Represent the following English sentences in first-order logic (with equality)

1. Only one prisoner can be in a cell
2. There is exactly one prisoner.

Use only the following predicates in your representation.

$Prisoner(p)$: p is a prisoner

$Cell(c)$: c is a cell

$In(p, c)$: Prisoner p is in cell c

B For each pair of atomic sentences below, say whether or not they are unifiable. If unifiable, give the most general unifier. Note that variables are in lowercase, whereas constant, predicate, and function symbols are in uppercase.

1. $P(x, F(x))$ and $P(G(y), y)$
2. $Q(y, y)$ and $Q(F(x), F(G(z))$

C Give a resolution proof to show that the following sentence is valid.

$$\exists y \forall x P(x, y) \Rightarrow \forall x \exists y P(x, y)$$

D Consider the following four propositions.

A: Adam is rich

B: Brian is rich

C: Christopher is rich

D: Daniel is rich

Express the following statement in conjunctive normal form in propositional logic.

"At least two of the four people (Adam, Brian, Christopher, Daniel) are rich"

3. Algorithms

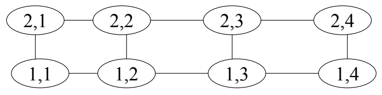
A Count the number of $*$, with respect to n, printed by each algorithm. Present your answer using the big theta Θ notation (no explanation is required). The answers have to be simplified. For example, suppose the correct bound is $\Theta(n^2)$, only partial credits will be awarded if you give $\Theta(n^2+n)$.

1.
```
for $i \leftarrow 1$ to $n$
    for $j \leftarrow 1$ to $n$
        if $i$ divides $j$, then print '$*$';
    end-of-for-loop
end-of-for-loop
```

2. The program $T(n)$, where the recursive function T is defined as follows.
```
function $T(i)$;
    if $i == 0$ then print '$*$';
    for $j \leftarrow 0$ to $(i-1)$
        $T(j)$; $T(j)$;
    end-of-for-loop
```

3. The program $S(n)$, where the recursive function S is defined as follows.
```
function $S(i)$;
    if $i > 0$
        $S(\lfloor 2i/3 \rfloor)$;   $S(\lfloor 2i/3 \rfloor)$;
    end-of-if
    for $j \leftarrow 0$ to $i\log(i)$
        print '$*$';
    end-of-for-loop
```

B The input of this problem is a weighted *2-tracked graph*, which is a undirected graph $G=(V,E)$ with $2n$ vertices. Each vertex is labeled as (i,j) where $i=1,2$, and $j=1,\ldots,n$. There is an edge between (i,j) to $(i,j+1)$ for every $i=1,2$ and $j=1,\ldots,n-1$. There is also an edge between $(1,j)$ and $(2,j)$ for all possible j's. Below is such a graph for $n=4$.



Furthermore, each vertex has a weight, which is a ***positive*** integer.

An *independent* set W is a subset of V, such that for any two vertices $u, v \in W$, there is no edge between u and v in the graph G. The total weight of a set W is the sum of the weights of all vertices in W; the total weight of the empty set is 0. There are many possible independent sets in G. Let us call the independent set with the largest total weight as the *maximum independent set.*

Give a $O(n)$ algorithm that, given a two-tracked graph, outputs the weight of its maximum independent set.
(Remark: Output the weight, which is a number. It is not necessary to find the maximum independent set.)
(Hint: You might want to use dynamic programming.)

4. Principles of Programming Languages

A The Fibonacci numbers are defined recursively by

$$Fib(n) \quad = \quad \begin{cases} 1 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ Fib(n-1) + Fib(n-2) & \text{if } n > 1 \end{cases}$$

We can straightforwardly implement the above in a C style language as follows.

```
int fib(int n){
 int f1, f2;
 if (n<=1) {return 1;}
 else { int f1=fib(n-1);
        int f2=fib(n-2);
        return f1+f2;}
 }
```

1. Explain the layout of `fib`'s activation record (also called frame).
2. Describe how the call stack of `fib` evolves during the execution of `fib(5)`.
3. Propose a compiler optimization to speed up the execution of `fib`.

B Consider the following *higher-order* function `reduce` written in a functional programming language like SML. A higher-order function can take in functions as arguments and return functions as return values. Thus,

```
fun reduce(f, nil, a) = a
 |   reduce(f, x::xs, a) = f x reduce(f,xs, a)
```

1. Make use of `reduce` to define a function `sum` to sum up a (non-empty) list of integers.
2. SML supports type inference. Hence, your next task is to give the most general type of `reduce`.
3. The Java programming language does not provide direct support for higher-order functions. However, Java does support objects via its class system. Your task is to define a general translation scheme from a simply-typed functional language (where function arguments can be functions themselves) to Java. It is sufficient to sketch the translation scheme by example.