University of Toronto, Faculty of Arts and Science

APRIL/MAY 2013 EXAMINATIONS

CSC236H1S

Professor Azadeh Farzan

Duration: 3 hours

Name: ______________________________

Student Number: ______________________________

Read the following before you start to work.

- Write your name and student number. Please write down your complete name (first name followed by last name) as it appears on the university records.
- This is a closed book exam. You are allowed a (possibly double-sided) **handwritten** 8.5×11 sheet of paper.
- Reminder: you need to get a mark of 35% or higher in this exam (21 marks in this case) to pass this course, regardless of your marks for the rest of the coursework
- You should have 18 pages including 6 problems. Do all work in the space provided. Ask the proctor if you need more paper.

Problem (1)	/8
Problem (2)	/7
Problem (3)	/9
Problem (4)	/14
Problem (5)	/12
Problem (6)	/10
Total	/60

Problem 1 A full binary tree is a binary tree in which every node has zero or two children (i.e. there is no node with exactly one child). Let L_n represent the number of leaves in a full binary tree with n nodes.

(a) (2 points) Draw all binary trees with 1, 3, and 5 nodes and determine the values of L_1, L_3 and L_5. Why didn't we ask for L_2 and L_4?

(b) (3 points) Write a recursive definition for L_n.

(c) (3 points) Solve the recursive definition to find a closed form solution for L_n (no proof required, however by "solving", we mean something more than guesswork).

Continue with the solution of the problem, if you need more space ...

Problem 2

(a) (3 points) Prove, using closure properties of regular languages, that if L is regular, then so is $L' = \{xy \mid x \in L \wedge y \notin L\}$. Note that proofs that are not based on closure properties of regular languages will get no credit.

(b) (4 points) Let $M = (Q, \Sigma, \delta, q_0, F)$ be the DFA that accepts L (i.e. $L = L(M)$). By defining the components below, define a finite automaton $M' = (Q', \Sigma', \delta', q_0', F')$ that accepts L'. You are allowed to use ϵ transitions.

- $Q' =$
- $q_0' =$
- $F' =$
- Finally, to define δ', you have the option of defining it formally (like the components above) or drawing a diagram that clearly shows what δ' looks like. Briefly explaining your formula or your diagram (whichever you choose).

Continue with the solution of the problem, if you need more space ...

Problem 3 Consider the language L consisting of the set of all strings over $\{a, b, c\}$ that contain at least two consecutive a's but do not contain two consecutive b's.

(a) (4 points) Write a regular expression for L. Explain why your regular expression is correct by explaining what the various parts of it mean.

(b) (2 points) Propose a deterministic finite automaton that accepts L.

(c) (3 points) For each state q in your finite automaton, describe the set of strings $\{x \mid \delta^*(q_0, x) = q\}$ that cause the finite automaton to reach that state starting from the initial state q_0.

Continue with the solution of the problem, if you need more space ...

Problem 4 Consider the following algorithm that, given positive integers a and b, computes the quotient q and remainder r of a divided by b (i.e. $a = bq + r$ where $q \geq 0$ and $0 \leq r < b$):

```
algorithm DIV(a, b)
    p := 1
    s := b
    while s ≤ a
        s := 2 × s
        p := 2 × p
    q := 0
    r := a
    while s ≠ b
        s := s div 2
        p := p div 2
        if r ≥ s
            r := r − s
            q := q + p
```

(a) (2 points) Write pre and post conditions, and a precise statement of what it means for this algorithm to be totally correct.

(b) (2 points) Write an invariant relating s and p at the entry point of both while loops. Briefly justify why these are loop invariants.

(c) (7 points) Prove that $a = qb + r$ combined with your answer from part (b) is an inductive loop invariant of the second while loop.

Continue with the solution of the problem, if you need more space ...

(d) (3 points) We are interested in a complexity measure for this algorithm, in the form of the number of iterations of the first and the second loops, as a parameter of the input values a and b. Let R be the function that represents the total number of iterations (of both loops). Regardless of what a and b are specifically, the number of iterations of the loops depends on the value a/b. Let $k = \lceil a/b \rceil$. What is $R(k)$ for an arbitrary value k? Justify your answer properly, but you do not need to prove your statement correct.

Continue with the solution of the problem, if you need more space ...

Problem 5 (12 points) Call a regular expression $\emptyset$-free if it has no occurrences of $\emptyset$. Here are recursive definitions for the set of regular expressions $\mathcal{R}$ and the set of $\emptyset$-free regular expressions $\mathcal{R}_\emptyset$:

$$\mathcal{R} : \begin{cases} \text{Basis:} & \begin{cases} \epsilon \\ a \quad \forall a \in \Sigma \\ \emptyset \end{cases} \\ \text{Induction Step:} & \begin{cases} r_1 + r_2 & \forall r_1, r_2 \in \mathcal{R} \\ r_1 r_2 & \forall r_1, r_2 \in \mathcal{R} \\ r_1^* & \forall r_1 \in \mathcal{R} \end{cases} \end{cases} \qquad \mathcal{R}_\emptyset : \begin{cases} \text{Basis:} & \begin{cases} \epsilon \\ a \quad \forall a \in \Sigma \end{cases} \\ \text{Induction Step:} & \begin{cases} r_1 + r_2 & \forall r_1, r_2 \in \mathcal{R} \\ r_1 r_2 & \forall r_1, r_2 \in \mathcal{R} \\ r_1^* & \forall r_1 \in \mathcal{R} \end{cases} \end{cases}$$

Prove (by induction) that for every regular expression r, if $L(r) \neq \emptyset$, then there exists an equivalent $\emptyset$-free regular expression r'. Or, more formally:

$$\forall r \in \mathcal{R}, [(L(r) \neq \emptyset) \implies (\exists r' \in \mathcal{R}_\emptyset,\ L(r) = L(r'))]$$

Continue with the solution of the problem, if you need more space ...

Continue with the solution of the problem, if you need more space ...

Problem 6 For all strings $u, v \in \Sigma^*$, we say that $v = u^R$ if

$$|u| = |v| \ \wedge \ \forall\, 0 \leq i < |u|,\ u[i] = v[|u| + 1 - i].$$

For example, $abcde = (edcba)^R$. Consider the algorithm below that reverses a string $u \in \Sigma^*$:

```
algorithm REV(u)
    l := |u|
    if l ≤ 1
        return u
    m := l div 2
    v := REV(u[1...m])
    w := REV(u[m+1...|u|])
    return wv
```

where $u[i \dots j]$ is the substring of u from position i to position j (both inclusive). We also assume that strings are indexed from 1 to the length of the string. The goal is to prove that algorithm REV correctly reverses a string.

(a) (2 points) Write pre and post conditions for *REV* and state a precise statement for correctness of REV.

(b) (8 points) Prove (by induction) that REV is correct in accordance with the statement from part (a).

Continue with the solution of the problem, if you need more space ...