University of Toronto, Faculty of Arts and Science

December 2015 EXAMINATIONS

CSC236H1F

Professor Azadeh Farzan

Seyed Amir Hejazi

(December 15, 2015: 9am-12pm)

Duration: 3 hours

Name: _____

Student Number: _____

Lecture Session (circle one): _ Morning _ Afternoon _ Evening _

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. Circle the section in which you are **registered**.
- This is a closed book exam. You are allowed a double-sided **handwritten** 8.5 × 11 sheet of paper. No other aids allowed (including calculators).
- Reminder: you need to get a mark of 35% or higher in this exam (18.9 marks in this case) to pass this course, regardless of your marks for the rest of the coursework.
- You should have 16 pages including 6 problems. Do all work in the space provided. Ask for more paper if needed.
- We provided you with ample blank space after each problem, so that you have some space for scratch work. If you do not erase your scratch work, clearly cross it out so that it is not considered for marking. The space provided for each solution is large enough, but not indicative of how long the correct answer is.
- You can find the statement of the Master Theorem on the last page.

Problem (1)	/6
Problem (2)	/9
Problem (3)	/17
Problem (4)	/5
Problem (5)	/7
Problem (6)	/11
Total	/55

Problem 1 Consider the following algorithm:

```
DAC(n)
1   If (n = 1)
        return true
2   DAC(⌊n/2⌋)
3   i = 1
4   while (i <?)
5       DAC(⌊n/2⌋)
6       i = i + 1
7   finish(n)
```

where it is assumed that **finish**(n) runs in time $\Theta(n^2)$.

(a) (3 points) Find a positive **constant** to place instead of "?" in the code, so that the asymptotic running time of the algorithm **DAC** becomes $\Theta(n^2)$. Justify your choice through the Master theorem.

Note: there may be more than one correct answer to this question. It suffices that you give us one correct answer.

(b) (3 points) Repeat part (a), but this time to achieve the asymptotic running time of $\Theta(n^3)$.

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

Problem 2 Consider an array A with numerical elements. The following algorithm recursively finds and returns the average over all elements in $A[b \ldots e]$.

```
Average(A, b, e)
1   If (b = e)
2       return A[b]
3   m = ⌊(b + e)/2⌋
4   x = Average(A, b, m)
5   y = Average(A, m + 1, e)
6   return ((m−b+1)×x+(e−m)×y) / (e−b+1)
```

(a) (2 points) Write the appropriate precondition and postcondition that specify the correctness of this function.

(b) (7 points) Prove that the function is correct by showing that the your specification from part (a) is inductive. Don't forget about termination.

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

Problem 3 Let us fix the alphabet $\Sigma = \{a, b, c\}$. Solve the following 4 problems:

(a) (4 points) Write a regular expression r such that $L_1 = L(r)$ for the following language

$$L_1 = \{w \in \Sigma^* | \text{ There are no two consecutive } a\text{'s anywhere in } w\}$$

Briefly explain why your regular expression is correct.

(b) (4 points) Draw a deterministic finite automaton that recognizes the following language:

$L_2 = \{w \in \Sigma^* |$ There are no two consecutive a's, no two consecutive b's, and no two consecutive c's anywhere in $w\}$

(c) (5 points) Formally explain your DFA design from part (b), by giving state invariants for all states of your DFA. There is no need to prove the invariants, but invariant definitions need to be precise and perfect.

(d) (4 points) Draw an NFA that recognizes the following language (no more than 6 states):

$$L_3 = \{w \in \Sigma^* | \text{ the last letter of } w, \text{ does not appear anywhere else in } w\}$$

briefly explain your NFA design. Note that we are not implying that the correct answer has exactly 6 states.

Problem 4 (5 points) Use closure properties of regular languages to argue that if L_1, L_2 and L_3 are regular (all three over the alphabet Σ), then so is L defined below:

$$L = \{w \in \Sigma^* | \ w \text{ belongs to at least two out of three languages } L_1, L_2, L_3\}$$

Note that any argument other than the use of closure properties will not be accepted.

Problem 5 (7 points) Consider the following program and the given precondition. Prove that if the precondition holds at the beginning of the program, then the loop always terminates.

PreCondition: arrays A and B are sorted, and $\exists k, l : A[k] = B[l]$

```
1   i = 0
2   j = 0
3   while (A[i] ≠ B[j])
4       if (A[i] < B[j])
5           i = i + 1
6       else if
7           j = j + 1
```

Note that indices l, k are implicitly assumed to be within range in both arrays A and B in the precondition (for brevity/ease of read); the longer explicit version of it is:

$$\exists k, l : (0 \leq k < |A|) \wedge (0 \leq l < |B|) \wedge (A[k] = B[l])$$

Be careful to prove any invariants of the loop that you **may** use in your termination proof.

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

Problem 6 Consider an array A, which contains integer elements. The goal of the following algorithm is to *separate* the elements of A into evens and odds so that all even elements come first, followed by all odd elements. The algorithm makes use of an auxiliary operation $\mathbf{swap}(A, i, j)$ that, as the name suggests, swaps the contents of the two array cells $A[i]$ and $A[j]$. For example, if $A[i] = 3$ and $A[j] = 5$, then after the the step $\mathbf{swap}(A, i, j)$, we will have $A[i] = 5$ and $A[j] = 3$. Naturally, if after the operation $\mathbf{swap}(A, i, i)$, the content of $A[i]$ (swapped with itself) does not change.

```
algorithm Parity(A)
1    b = 0
3    e =length(A) -1
4    while (b < e)
5         if (A[b] is even)
6              b = b + 1
7         else
8              swap(A, b, e)
9              e = e − 1
```

(a) (2 points) Write the appropriate formal precondition and postcondition that specify the correctness of this function.

(b) (2 points) Write an appropriate loop invariant that you can use to prove this function correct.

(c) (7 points) Prove that the function is partially correct by showing that the your specifications from parts (a) and (b) are inductive, or by providing a full induction proof. Declare clearly at the beginning which you choose to do.

(The code is repeated on this page for your benefit; so that you don't have to keep looking back and forth while writing your proof.)

```
algorithm Parity(A)
1    b = 0
3    e =length(A) -1
4    while (b < e)
5        if (A[b] is even)
6            b = b + 1
7        else
8            swap(A, b, e)
9            e = e − 1
```

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

Use this space for scratch work or if you need more room for any of the exam problems ...

Exam Cheat Sheet

Master Theorem:

$$T(n) = aT(n/b) + f(n), \text{ where } a \geq 1 \text{ and } b > 1$$

- If $c < \log_b a$ and $f(n) = \Theta(n^c)$ then $T(n) = \Theta(n^{\log_b a})$.
- If $c > \log_b a$ and $f(n) = \Theta(n^c)$ then $T(n) = \Theta(f(n))$.
- If $c = \log_b a$ and $f(n) = \Theta(n^c \log^k n)$ then $T(n) = \Theta(n^c \log^{k+1} n)$.

Another version of Master Theorem:

For $T(n) = a_1 T(n/b) + a_2 T(n/b) + f(n)$, if $f(n) = \Theta(n^d)$ for constant $d \geq 0$, and $a = a_1 + a_2$, then the recurrence $T(n)$ above has the asymptotic complexity:

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Short-cut solutions table:

INDUCTIVE EQUATION	$T(n)$
$T(n) = T(n-1) + bn^k$	$O(n^{k+1})$
$T(n) = cT(n-1) + bn^k$, for $c > 1$	$O(c^n)$
$T(n) = cT(n/d) + bn^k$, for $c > d^k$	$O(n^{\log_d c})$
$T(n) = cT(n/d) + bn^k$, for $c < d^k$	$O(n^k)$
$T(n) = cT(n/d) + bn^k$, for $c = d^k$	$O(n^k \log n)$