Computer Science CSC238
St. George Campus
Midterm Test #1
Examiner: James MacLean

Monday, February 22, 1999
University of Toronto
Day Section (L0101)

Last Name:

First Name(s):

Student number:

Tutor's name:

There are 4 questions for a total of 50 marks. The total time is 50 minutes. No aids are allowed. Answer directly on the paper.

Question 1.	/ 5
Question 2.	/10
Question 3.	/15
Question 4.	/20
Total	/50

1. [5 marks] Find the flaw in the following "proof" that all horses are the same colour.

Proof.

Let $P(n)$ be the predicate of natural numbers defined as
$P(n)$: "All horses in a set of n horses are the same colour".

BASIS: ($n = 1$) $P(1)$ is trivially true. (We ignore $n = 0$ since we are only interested in non-empty sets of horses.)

INDUCTION STEP: Let $n \geq 1$ be a natural number. Assume that $P(n)$ holds. We shall prove that $P(n+1)$ also holds.

Consider any $n+1$ horses. Number these horses as $1, 2, 3, \ldots, n, n+1$. Now, the first n of these horses must all have the same colour (since $p(n)$ holds via the induction hypothesis), and the last n must have the same colour also. Since these two subsets of the $n+1$ horses overlap, all $n+1$ must be the same colour.

Thus $P(n+1)$ holds, and $P(n)$ holds for all $n \geq 1$ by simple induction.

Answer: When $n + 1 = 2$, each subset has 1 element only, and the two subsets do not overlap, so the proof fails here.

2. [10 marks] For all $n \in \mathbb{N}$, prove that $2^{2n+1} + 1$ is divisible by 3. [**Definition:** An integer a is said to be divisible by another integer b if there exists an integer c such that $a = b \cdot c$.]

Answer:

(2 marks) Let $P(n)$ be the following predicate defined on the natural numbers:

$P(n)$: $2^{2n+1} + 1$ is divisible by three.

Basis: (2 marks) $n = 0$. The $2^{2n+1} + 1 = 2^1 + 1 = 3$ which is obviously divisible by 3.

Induction Step: For some arbitrary $m \in \mathbb{N}$ assume $P(m)$ holds, that is $2^{2m+1} + 1$ is divisible by 3. Prove that $P(m + 1)$ also holds.

$$\begin{aligned} 2^{2(m+1)+1} + 1 &= 2^{2m+3} + 1 \\ &= 2^2(2^{2m+1}) + 1 \\ &= 4(2^{2m+1} + 1 - 1) + 1 \\ &= 4(2^{2m+1} + 1) - 3 \end{aligned}$$

We have the sum of two terms, each of which is divisible by 3 (the first term via the induction hypothesis, the second term by the fact that -3 is obviously divisible by 3). Therefore $2^{2(m+1)+1} + 1$ is divisible by 3, and by induction $P(n)$ is true for all $n \in \mathbb{N}$.

For those who wish to be a bit more thorough, since $2^{2m+1} + 1$ is divisible by 3 (via the induction hypothesis), then there exists some q such that $3q = 2^{2m+1} + 1$. Thus we have $4(3q) - 3 = (4q - 1)3 = q'3$ where $q' = 4q - 1$. Thus $2^{2(m+1)+1} + 1 = q'3$ and is therefore divisible by 3.

2 marks for appropriate use of induction hypothesis (i.e. identifying it when it is used). 4 marks for correct arithmetic.

3. [15 marks]
Consider the following program.

```
FACT(n)
    x := 1;
    m := n;
    while (m ≠ 0) do begin
        x := x * m;
        m := m − 1;
    end while
    return (x)
END
```

This program calculates $n!$ (n factorial) for $n \in \mathbb{N}$.

(a) [7 marks] For this question, we use subscript i, $i \geq 0$, to denote the value of each variable at the point just before the test for $(m \neq 0)$, after i iterations have been completed.

Write an appropriate loop-invariant which will allow you to prove partial correctness in (b).

Answer: An appropriate loop invariant is

$$x_i = \frac{n!}{(n-i)!} \; .$$

We see that $x_0 = 1$, $x_1 = n$, $x_2 = n(n-1)$, *etc.* We also have $x_n = n!$, which is the desired result. We could try something like

$$x_i = \prod_{j=0}^{i} (n - j)$$

but it is harder to get this right for x_0.

(b) [4 marks] Prove the partial correctness of the algorithm.

Answer: Note that $m_i = n - i$. If the loop terminates, it does so when $m_i = 0$ which happens when $i = n$. When $i = n$, $x_n = \frac{n!}{0!} = n!$. Therefore, if the loop terminates, the value returned is $x_m = n!$ as required. It remains to prove the loop invariant.

Let $P(i)$ be the predicate on $i \in \mathbb{N}$ defined as

$$P(i) : x_i = \frac{n!}{(n-i)!}$$

Basis case: $i = 0$. $x_0 = 1$ by line 1 of the function.

Induction step: For some arbitrary $i > 0$ assume $P(i)$ holds, and the ith loop iteration exists. We must prove $P(i+1)$ holds (if the $(i+1)$th loop iteration exists).

$$\begin{aligned} x_{i+1} &= x_i m_i \\ &= x_i(n-i) \quad \text{[by definition of } m_i\text{]} \\ &= \tfrac{n!(n-i)}{(n-i)!} \quad \text{[by induction hypothesis]} \\ &= \tfrac{n!}{(n-(i+1))!} \,. \end{aligned}$$

Therefore, if the $(i+1)$th iteration exists, then $P(i+1)$ holds.

Therefore by induction $P(i)$ holds for all $i \in \mathbb{N}$ (if the ith loop iteration exists).

(c) [4 marks] Prove that the algorithm terminates.

Answer: Note that $m_i = n - i = m_{i-1} - 1$. Clearly the $\{m_i\}$ form a decreasing sequence. Assuming $n \in \mathbb{N}$, then $m_0 = n$, $m_1 = n - 1$, $m_2 = n - 2$, *etc.* and the $\{m_i\}$ are a decreasing sequence of natural numbers (since the loop terminates for $m_i = 0$ we never have a negative m_i), and by theorem 1.25 in the course notes the sequence is finite, thus the loop terminates.

4. [20 marks]
Consider the following program.

```
BinarySearch(A, f, l, key)
m := (f + l) div 2
if (A[m] = key) then return m
else if (f = l) then return −1
else if (A[m] > key) then return BinarySearch(A, f, m − 1, key)
else return BinarySearch(A, m + 1, l, key)
End
```

This algorithm performs a binary search for an element *key* in a **sorted** array A. If the element is found, its array indice is returned. If the element is not found, -1 is returned.

(a) [5 marks] Let $T(n)$ be the worst-case time complexity of BinarySearch(A, f, l, key) where $n = l - f + 1$. Give a recurrence relation for $T(n)$. [**Note:** You may treat both recursive calls as operating on data of size $(f + l)/2$ even though in the case $A[m] > key$ this is not strictly true.]

Answer: There is an error in the question: $(f+1)/2$ should have read $n/2$. Assuming worst case behaviour, at each step in the search the value to be found is always in one of the remaining sub-arrays, requiring recursion until the sub-array has only one element in it.

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\lceil n/2 \rceil) + 1 & n > 1 \end{cases}$$

Note the use of ceiling instead of floor ... again we are looking at worst case behaviour. We can use an arbitrary constant (k) instead of 1, but it doesn't change the asymptotic order of the complexity.

(b) [5 marks] Use "repeated substitution" to guess a solution for the recurrence in part (a) (i.e. guess a closed formula for $T(n)$) when n is a power of 2. Show your work.

Answer: We have $n = 2^k$, or $k = \log_2 n$. Use repeated substitution we get

$$\begin{aligned}
T(n) &= T(n/2) + 1 && \text{[assuming n ¿ 1]} \\
&= (T(n/4) + 1) + 1 && \text{[via recurrence relation from (a), n/2 ¿ 1]} \\
&= T(n/2^2) + 2 \\
&= (T(n/8) + 1) + 2 && \text{[assuming n/4 ¿ 1]} \\
&= T(n/2^3) + 3 \\
&\ldots \\
&= T(n/2^k) + k \\
&= k + 1 && [\text{since } n = 2^k] \\
&= \log_2(n) + 1
\end{aligned}$$

(c) [6 marks] Prove by induction that your formula in part (b) holds when n is a power of 2.

Answer: Let $n = 2^k$ for some $k \in \mathbb{N}$. Let $P(k)$ be the predicate defined on the natural numbers as

$$P(k) : T(n) = \log_2 2^k + 1 = k + 1$$

Base case: $k = 0$. Then $\log_2 2^0 + 1 = \log_2 1 + 1 = 1 = T(1)$

Induction step: Assume $P(k)$ holds for some arbitrary $k \in \mathbb{N}$. We must prove $P(k+1)$ also holds. That is, we must prove that $T(2n) = \log_2 2^{k+1} + 1 = k + 2$.

Let $n_k = 2^k$ and $n_{k+1} = 2^{k+1}$. We have

$$
\begin{aligned}
T(n_{k+1}) &= T(\tfrac{n_{k+1}}{2}) + 1 && \text{[via recurrence relations, since } k+1>0\text{]}\\
&= T(n_k) + 1 \\
&= (log_2 2^k + 1) + 1 && \text{[induction hypothesis]}\\
&= (k+1) + 1 \\
&= k + 2 && \text{[as required]}
\end{aligned}
$$

Therefore, by induction, $P(k)$ holds for all $k \in \mathbb{N}$.

(d) [4 marks] This part is **not** restricted to the case of n being a power of 2. Assume $T(n)$ is non-decreasing. Give a Θ estimate for $T(n)$. Justify.

Answer: The recurrence relation from (a) fits the general form for divide and conquer recurrences with $a = 1$, $b = 2$, $c = 1$ and $d = 0$. Since $a = b^d$ and $T(n)$ is non-decreasing, we can invoke Theorem 2.8 to show that $T(n) \in \Theta(\log_2 n)$.