Name:
utorid:
U of T email:

Please read the following guidelines carefully!

- Please write your name on both the front and back of this exam.
- This examination has 4 questions. There are a total of 8 pages, DOUBLE-SIDED.
- Answer questions clearly and completely. Provide justification unless explicitly asked not to.
- All formulas must have negations applied directly to propositional variables or predicates.
- In your proofs, you may always use definitions of predicates from the course. You may not use any external facts about rates of growth, divisibility, primes, or greatest common divisor unless you prove them, or they are given to you in the question.

Take a deep breath.
This is your chance to show us
How much you’ve learned.

Good luck!
1. [5 marks] Induction. Recall the definition of divisibility:

\[ j \mid k : \exists i \in \mathbb{Z}, k = ji, \text{ for } j, k \in \mathbb{Z} \]

Now prove the following statement using induction on \( n \):

\[ \forall n \in \mathbb{N}, 5 \mid 4^{2n} - 1 \]

Solution

Proof (induction on \( n \)): Define predicate \( P(n) : 5 \mid 4^{2n} - 1 \).

Base case: Let \( i = 0 \), and note that \( 5i = 0 = 4^{2(0)} - 1 \), which verifies \( P(0) \).

Inductive step: Let \( n \in \mathbb{N} \). Assume \( P(n) \), that is \( \exists i_n \in \mathbb{Z}, 5i_n = 4^{2n} - 1 \). Let \( i_n \) be such a value, and let \( i_{n+1} = 16i_n + 3 \). I will prove \( P(n + 1) \):

\[
4^{2(n+1)} - 1 = 4^{2n+2} - 1 = 4^2(4^{2n} - 1) + 15
= 16(5i_n) + 5(3) \quad \text{(by Inductive Hypothesis)}
= 5(16i_n + 3) = 5i_{n+1}
\]
2. [5 marks] Properties of Big-Oh. Recall the following definitions:

- For all functions $f, g : \mathbb{N} \rightarrow \mathbb{R}^\geq 0$, we say $f \in O(g)$ when:
  \[ \exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \leq cg(n) \]

- For any real number $x$, $[x]$ is the smallest integer that is no smaller than $x$, and we may use the following characterization of $[x]$:
  \[ x \leq [x] < x + 1 \]

Define the function $\lfloor f \rfloor(n)$ as $[f(n)]$.

- Function $f$ eventually dominates $1$ if:
  \[ \exists n_1 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_1 \Rightarrow f(n) \geq 1 \]

Use these definitions (you may not use any of the properties of big-Oh from the course notes) to prove that if $f \in O(g)$ and $f$ eventually dominates $1$, then $\lfloor f \rfloor \in O(g)$. Begin by writing a statement, in predicate logic, of what you aim to prove.

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
</table>
| **Claim:**
| $\forall f, g \in \mathbb{N} \rightarrow \mathbb{R}^\geq 0, [f \in O(g) \land (\exists n_1 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_1 \Rightarrow f(n) \geq 1)] \Rightarrow [f] \in O(g)$ |
| **Proof:** Let $f, g \in \mathbb{N} \rightarrow \mathbb{R}^\geq 0$. Assume $f \in O(g)$, that is $\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \leq cg(n)$. Let $c$ and $n_0$ be such values. Also assume $\exists n_1 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_1 \Rightarrow f(n) \geq 1$, and let $n_1$ be such a value. Let $n_2 = \max(n_0, n_1)$ and $c_1 = 2c$. I will show that $\forall n \in \mathbb{N}, n \geq n_2 \Rightarrow [f(n)] \leq c_1g(n)$.

Let $n \in \mathbb{N}$ and assume $n \geq n_2$. Then

$\lfloor f(n) \rfloor \leq f(n) + 1$ (characterization of $[x]$)
\[ \leq f(n) + f(n) = 2f(n) \quad \text{(since } n \geq n_1) \]
\[ \leq 2cg(n) \quad \text{(since } n \geq n_0) \]
\[ = c_1g(n) \quad \blacksquare \]

never, ever, write below this line...
3. [6 marks] Worst-case runtime

Consider the following algorithm:

```python
def algorithm(L):
    # assume L is a non-empty list of 0s and 1s
    n = len(L)
    parity = L[0]
    last_switch = 0
    for i in range(n):
        # loop 1
        if parity != L[i]:
            last_switch = i
        parity = L[i]
    for j in range(last_switch):
        # loop 2
        for k in range(j):
            # loop 3
            print("beep!")
```

Define \( n = \text{len}(L) \) and \( WC(n) \) as the worst-case runtime function of \( \text{algorithm} \). You may find the following formula useful:

\[
\sum_{i=0}^{m} i = \frac{m(m+1)}{2}
\]

(a) [4 marks] Find, and prove, a tight upper bound on \( WC(n) \). By “tight” we mean that if you choose \( f \) so that \( WC \in \mathcal{O}(f) \) you should be convinced (but no need to prove) that \( WC \in \Omega(f) \) also. Begin by writing a statement, in predicate logic, of what you aim to prove.

Solution

Claim: Define \( I_{\text{algorithm},n} = \{ \text{lists of length } n \text{ consisting only of 0s and 1s} \} \) and \( RT(x) \): “steps to execute \( \text{algorithm}(x) \)” for \( x \in I_{\text{algorithm},n} \). Let \( U(n) = 7n^2 \). I will show that \( U(n) \) is an upper bound on \( WC_{\text{algorithm}}(n) \) by showing that:

\[
\forall n \in \mathbb{N}, \forall x \in I_{\text{algorithm},n}, RT(x) \leq 7n^2
\]

Proof: Let \( n \in \mathbb{N} \) and \( x \in I_{\text{algorithm},n} \). Then \( RT(x) \) costs at most:

- 3 steps for lines 3–5,
- 4 steps for lines 6–9 for each \( i \), or \( 4n \) altogether (if we count each line as a step)
- \((n - 1)^2 = n^2 - 2n + 1\) steps for lines 11–13, since \( j = \text{last_switch} \) is at most \( n - 1 \) and \( k \) is never more than \( j \), so the inner loop iterates at most \( n - 1 \) times for each \( j \), and there are no more than \( n - 1 \) iterations of the outer loop
In total there are no more than:

\[ n^2 - 2n + 1 + 4n + 3 = n^2 + 2n + 4 \leq 7n^2 \quad (\text{since } n \geq 1) \]

(b) [2 marks] Describe an input family for algorithm whose runtime is in big-Theta of the upper bound from the previous part. Explain your conclusion. No proof is necessary.

**Solution**

**Sample solution:** Consider the family of lists of the form \(x_n = [0, 0, \ldots, 0, 1]\). Then `last_switch` has value \(n - 1\) on line 10. Loop 3 takes \(j\) steps for each fixed \(j\), where \(j\) runs from 0 to \(n - 2\), so lines 11–13 perform:

\[
\sum_{j=0}^{n-2} j = \frac{(n - 2)(n - 1)}{2} = \frac{n^2 - 3n + 2}{2}
\]

...steps, which is at least \(n^2/4\) provided \(n\) is at least three. This is in \(\Omega\) of \(U(n)\), and the previous part showed it was in big-Oh of \(U(n)\). The steps contributed by lines 1–10 need not be considered, since they will not change this \(\Omega\) bound.

4. [3 marks] Describe an input family whose runtime is in \(O(n)\). Explain your conclusion. No proof is necessary.

**Solution**

**Sample solution:** Consider the family of lists with only 0s as elements. Then lines 11–13 contribute no steps, since `last_switch` has value 0 on line 10, so the runtime consists of 3 steps for lines 3–5 and \(4n\) steps for lines 6–9, for a total of \(4n + 3\) steps, which is in \(O(n)\).
This page is left nearly blank for rough work. If you want work on this page to be marked, please indicate this clearly at the location of the original question.
This page is left nearly blank for rough work. If you want work on this page to be marked, please indicate this clearly at the location of the original question.
Name:

<table>
<thead>
<tr>
<th>Question</th>
<th>Grade</th>
<th>Out of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Q4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>19</strong></td>
<td></td>
</tr>
</tbody>
</table>