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:

$j \mid k$: $\exists i \in \mathbb{Z}, k = ji$, 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 $P(n+1)$:

$$
\begin{aligned}
4^{2(n+1)} - 1 = 4^{2n+2} - 1 &= 4^2(4^{2n} - 1) + 15 \\
&= 16(5i_n) + 5(3) \qquad \text{(by Inductive Hypothesis)} \\
&= 5(16i_n + 3) = 5i_{n+1} \qquad \blacksquare
\end{aligned}
$$

---

never, **ever**, write below this line...

2. **[5 marks] Properties of Big-Oh.** Recall the following definitions:

  - For all functions $f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$, we say $f \in \mathcal{O}(g)$ 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, $\lceil x \rceil$ is the smallest integer that is no smaller than x, and we may use the following characterization of $\lceil x \rceil$:
  $$x \leq \lceil x \rceil < x + 1$$
  Define the function $\lceil f \rceil (n)$ as $\lceil f(n) \rceil$.

  - 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 $f \in \mathcal{O}(g)$ and f eventually dominates 1, then $\lceil f \rceil \in \mathcal{O}(g)$. Begin by writing a statement, in predicate logic, of what you aim to prove.

Solution

Claim:
$$\forall f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}, \left[f \in \mathcal{O}(g) \wedge \left(\exists n_1 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_1 \Rightarrow f(n) \geq 1 \right) \right] \Rightarrow \lceil f \rceil \in \mathcal{O}(g)$$

Proof: Let $f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$. Assume $f \in \mathcal{O}(g)$, 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 $\forall n \in \mathbb{N}, n \geq n_2 \Rightarrow \lceil f(n) \rceil \leq c_1 g(n)$.

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

$$
\begin{aligned}
\lceil f(n) \rceil \quad &< \quad f(n) + 1 \quad &&\text{(characterization of } \lceil x \rceil \text{)} \\
&\leq \quad f(n) + f(n) = 2f(n) \quad &&\text{(since } n \geq n_1 \text{)} \\
&\leq \quad 2cg(n) \quad &&\text{(since } n \geq n_0 \text{)} \\
&= \quad c_1 g(n) \quad \blacksquare
\end{aligned}
$$

---

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 $n = \mathrm{len}(L)$ and $WC(n)$ as the worst-case runtime function of `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 $\mathcal{I}_{\text{algorithm},n} = \{\text{lists of length } n \text{ consisting only of 0s and 1s}\}$ and $RT(x) :$ "steps to execute `algorithm(x)`" for $x \in \mathcal{I}_{\text{algorithm},n}$. Let $U(n) = 3n^2$. I will show that $U(n)$ is an upper bound on $WC_{\text{algorithm}}(n)$ by showing that:
   >
   > $$\forall n \in \mathbb{N}, \forall x \in \mathcal{I}_{\text{algorithm},n}, RT(x) \le 7n^2$$
   >
   > **Proof:** Let $n \in \mathbb{N}$ and $x \in \mathcal{I}_{\text{algorithm},n}$. 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=(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:

$$
\begin{aligned}
n^2 - 2n + 1 + 4n + 3 \; &= \; n^2 + 2n + 4 \\
&\leq \; 7n^2 \quad (\text{ since } n \geq 1)
\end{aligned}
$$

(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 $x_n = [0, 0, ..., 0, 1]$. 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 Ω 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 Ω bound.

---

4. **[3 marks]** Describe an input family whose runtime is in $\mathcal{O}(n)$. 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_switchh` 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 $\mathcal{O}(n)$.

---

never, **ever,** write below this line...

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.*

never, **ever,** write below this line...

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.*

*never, **ever,** write below this line...*

Name:

Question	Grade	Out of
Q1		5
Q2		5
Q3		6
Q4		3
Total		19

never, **ever,** write below this line...