1. (12 points) Recall:

**Definition 6.1.** Let \( f, g : \mathbb{N} \to \mathbb{R}^+ \). We say that \( f \) is of order \( g \), written \( f(n) \in O(g(n)) \),
if there exist \( N \in \mathbb{N} \) and \( C \in \mathbb{R} \) such that for all \( n \geq N \), \( f(n) \leq C \cdot g(n) \).

(a) Find witnesses \( N \) and \( C \) that show \( 7n + 5 \in O(n) \).

\[
\begin{align*}
\{ & 7n \leq 7n \quad \text{if } n \geq 1, \\
& 5 \leq n \quad \text{if } n \geq 5 \\
& 7n + 5 \leq 8n \quad \text{if } n \geq 5 \}
\end{align*}
\]

\( N = 5 \)

\( C = 8 \)

(b) Find witnesses \( N \) and \( C \) that show \( 4n^2 + 3n + 1 \in O(n^2) \).

\[
\begin{align*}
\{ & 4n^2 \leq 4n^2 \quad \text{if } n \geq 1, \\
& 3n \leq n^2 \quad \text{if } n \geq 3 \\
& 1 \leq n^2 \quad \text{if } n \geq 1 \\
& 4n^2 + 2n + 1 \leq 4n^2 \quad \text{if } n \geq 3 \}
\end{align*}
\]

\( N = 3 \)

\( C = 4 \)

(c) Prove: If \( f_1(n) \in O(g_1(n)) \) and \( f_2(n) \in O(g_2(n)) \) then \( f_1(n) \times f_2(n) \in O(g_1(n) \times g_2(n)) \).

- Since \( f_1(n) \in O(g_1(n)) \) there exist \( N_1 \) and \( C_1 \) such that for all \( n \geq N_1 \), \( f_1(n) \leq C_1 \cdot g_1(n) \).
- Since \( f_2(n) \in O(g_2(n)) \) there exist \( N_2 \) and \( C_2 \) such that for all \( n \geq N_2 \), \( f_2(n) \leq C_2 \cdot g_2(n) \).
- All quantities are positive, so we can multiply inequalities to get \( f_1(n) \times f_2(n) \leq C_1C_2 \cdot g_1(n) \times g_2(n) \) for all \( n \geq N_1, N_2 \).
- Thus, \( f_1f_2 \in O(g_1g_2) \) with witnesses \( N = \max(N_1, N_2) \) and \( C = C_1C_2 \).
2.  (12 points) Analysis of recursive “divide and conquer” algorithms may yield recursive performance estimates like $T$, to the right. To show that $T(n) \in O(n \log n)$ it is easier to restrict $n$ to be a power of two:

Prove: For all $n \in \mathbb{N}$, $T(2^n) = 2^n(a + bn)$.

(BASE CASE) $T(2^0) = T(1) = a = 2^0 \cdot (a + 0 \cdot b)$

(INDUCTION STEP) Assume $T(2^k) = 2^k(a + bk)$. Then

$$T(2^{k+1}) = 2 \left[ T\left( \frac{2^{k+1}}{2} \right) \right] + 2^{k+1}b$$

$$= 2 \left[ T(2^k) \right] + 2^{k+1}b$$

$$\overset{IH}{=} 2 \left[ 2^k(a + bk) \right] + 2^{k+1}b$$

$$= 2 \left[ 2^k a + 2^k bk \right] + 2 \cdot 2^k b$$

$$= 2 \left[ 2^k a + 2^k bk + 2^k b \right]$$

$$= 2^{k+1}(a + b(k + 1))$$

as needed. This completes the induction.  \[ \square \]
3. (10 points) Recall that a substitution, \( F[U_1, \ldots, U_n] \) denotes the sentence obtained by simultaneously replacing all occurrences of variables \( v_i \) by the corresponding words \( U_i \) in formula \( F \).

For \( F \equiv x + (y - z) + wx \), write the results of the following substitutions:

(a) \( F[a, b][x, y] \) \( \equiv \) \((a + (b - z)) + wa\)

(b) \( F[b][q] \) \( \equiv \) \((x + (y - z)) + wx\)

(c) \( F[y, b][x, y] \) \( \equiv \) \((y + (b - z)) + wy\)

(d) \( F[y][b][x, y] \) \( \equiv \) \((b + (b - z)) + wb\)

(e) \( F[y-w][x][y-z][w] \) \( \equiv \) \((y - (y - z)) + (y - z) + (y - z)(y - w)\)

4. (10 points) Let \( A \) be the set of alphabetic characters, \( A = \{a, b, \ldots, z\} \) and let \( # \) and \$ be two operation symbols. Define a infix language in which

(a) \$ takes precedence over \#,

(b) \$ associates to the right, and

(c) \# associates to the left.

For instance, \( a \$ b \$ c # d # e \$ f \$ g \) would be parsed as

You may—but are not required to—use Backus-Naur notation (BNF).

\[
\begin{align*}
F \in (A \cup \{\$\})^+ \\
&1. \quad A \subseteq F \\
&2. \quad x \in A \land v \in F \Rightarrow x \$ v \in F \\
&3. \quad \text{nothing else}
\end{align*}
\]

\[
\begin{align*}
E \in (F \cup \{\#\})^+ \\
&1. \quad F \subseteq E \\
&2. \quad u \in E \land v \in F \Rightarrow u \# v \in F \\
&3. \quad \text{nothing else}
\end{align*}
\]
5. (16 points) The language \( L \) and functions \( I, A, R \) and \( T \), defined below, are the same as in Section 3.6.

\[
L \subseteq \{a, b, \bullet\}^+ \\
\begin{array}{l}
1. \quad \bullet \in L \\
2a. \quad u \in L \Rightarrow au \in L \\
2b. \quad u \in L \Rightarrow bu \in L \\
3. \quad \text{nothing else}
\end{array}
\]

\[
I: L \rightarrow L \\
\begin{array}{l}
1. \quad I(\bullet) = \bullet \\
2a. \quad I(au) = bI(u) \\
2b. \quad I(bu) = aI(u)
\end{array}
\]

\[
R: L \rightarrow L \\
\begin{array}{l}
1. \quad R(\bullet) = \bullet \\
2a. \quad R(au) = A(R(u), a\bullet) \\
2b. \quad R(bu) = A(R(u), b\bullet)
\end{array}
\]

\[
A: L^2 \rightarrow L \\
\begin{array}{l}
1. \quad A(\bullet, v) = v \\
2a. \quad A(bu, v) = bA(u, v) \\
2b. \quad A(au, v) = aA(u, v)
\end{array}
\]

\[
T: L^2 \rightarrow L \\
\begin{array}{l}
1. \quad T(\bullet, v) = v \\
2a. \quad T(au, v) = T(u, av) \\
2b. \quad T(bu, v) = T(u, bv)
\end{array}
\]

(a) Indicate which of the following are true:

\[
\begin{array}{c|c|c|c}
i) & T & \bullet \in L & F \\
ii) & F & a \in L & F \\
iii) & T & A(R(ba\bullet), a\bullet) = A(A(R(\bullet), b\bullet), a\bullet) & F \\
vii) & T & I(\psi) = \bullet & F \\
viii) & T & I(u) = u \begin{bmatrix} a \ b \\ b \ a \end{bmatrix} & F \\
\end{array}
\]

(b) Use structural induction to prove: For all \( u \in L \), \( A(u, \bullet) = u \).

**BASE CASE.** \( A(\bullet, \bullet) \stackrel{A.1}{=} \bullet \).

**INDUCTION:** Assume \( IH \equiv A(u, \bullet) = u \).

\[
A(au, \bullet) \stackrel{A.2a}{=} a \ A(u, \bullet) \stackrel{IH}{=} au
\]

Similarly, for \( bu \),

\[
A(bu, \bullet) \stackrel{A.2b}{=} b \ A(u, \bullet) \stackrel{IH}{=} bu
\]
6. (20 points) This question has four parts on this and the next page.

The language $\text{Tree}$, defined below, is a symbolic representation of binary tree structures.

$$\text{Tree} \subseteq \{N, L, (, )\}^+$$

1. $L \in \text{Tree}$
2. if $t_1, t_2 \in \text{Tree}$ then $(N^*t_1^*t_2^*) \in \text{Tree}$
3. nothing else

(a) Indicate which of these sentences are words in the language $\text{Tree}$?

(i) $Y^*L$
(ii) $Y^*N^*L^*L$
(iii) $Y^*N^*(N^*L^*L^*)^*L$
(iv) $N^*(L)$
(v) $N^*(N^*N^*L)$
(vii) $N^*(N^*(N^*L^*L^*)^*(N^*L^*))^*$

(b) The size of any word is be number of characters it contains. Fill in the missing information to define a recursive function $\text{size}: \text{Tree} \rightarrow \mathbb{N}$ that gives the size of any word $t \in \text{Tree}$.

1. $\text{size}(L) = 1$
2. $\text{size}(N^*t_1^*t_2^*) = 3 + \text{size}(t_1) + \text{size}(t_2)$

Full credit if you entered 1 instead of 3 above.

(c) The depth of a tree is defined to be one plus the length of the longest path from its root to a leaf. Define a recursive function $\text{depth}: \text{Tree} \rightarrow \mathbb{N}$ that gives the depth of the tree it represents.

$$\text{depth}(L) = 1$$
$$\text{depth}(N^*t_1^*t_2^*) = 1 + \max(\text{depth}(t_1), \text{depth}(t_2))$$
(d) Prove by induction that the size of a tree is exponential in its depth. That is, for some constant \( C \), and for any \( t \in \text{tree} \), \( \text{size}(t) < C \cdot 2^d \), where \( d = \text{depth}(t) \).

**Comment.** This is way too tricky to be a test question. Ample credit was given for any reasonable attempt. You had to answer parts (b) and (c) correctly to have much chance at a proof. If \( \text{size}(t) \) were defined to be the number of \( N \) s and \( L \) s, it is pretty easy to prove that \( \text{size}(t) \leq 2^{\text{depth}(t)} - 1 \). The purpose of \( C \) is to account for the parentheses. But the theorem as stated does not support an induction. Instead, one has to prove something stronger:

**Lemma.** For all \( t \in \text{tree} \), \( \text{size}(t) \leq 3(2^d - 1) \), where \( d = \text{depth}(r) \)

**Proof.**

**Base case.** \( \text{size}(L) = 1 \leq 3 \cdot 1 = 3(2^1 - 1) = 3(2^{\text{depth}(L)} - 1) \)

**Induction.** Let \( t = (N \ t_1 \ t_2) \) and \( s = \text{size}(t) \quad d = \text{depth}(t) = 1 + \max(d_1, d_2) \)
\[
\begin{align*}
  s_1 &= \text{size}(t_1) \quad d_1 = \text{depth}(t_1) \\
  s_2 &= \text{size}(t_2) \quad d_2 = \text{depth}(t_2)
\end{align*}
\]
Assume by induction that
\[
\begin{align*}
  s_1 &\leq 3(2^{d_1} - 1) \\
  s_2 &\leq 3(2^{d_2} - 1)
\end{align*}
\]
Let \( \tilde{d} = \max(d_1, d_2) \). Adding the inequalities above, we get
\[
s_1 + s_2 \leq 3(2^{\tilde{d}} - 1) + 3(2^{\tilde{d}} - 1) = 3(2^{\tilde{d}+1} - 2) = 3(2^{\tilde{d}} - 2)
\]
where \( d = \text{depth}(t) \), as defined above. Hence,
\[
\begin{align*}
  \text{size}(t) &= 3 + s_1 + s_2 \\
  &\leq 3 + 3(2^{\tilde{d}} - 2) \quad (\text{It is at this point that we learn } C \text{ must equal } 3) \\
  &= 3 + 3 \cdot 2^{\tilde{d}} - 6 \\
  &= 3 \cdot 2^d - 3 \\
  &= 3(2^d - 1)
\end{align*}
\]
This completes the induction step.

**Corollary.** For some constant \( C \) and for all \( t \in \text{tree} \), \( \text{size}(t) < C \cdot 2^d \), where \( d = \text{depth}(t) \).

**Proof.** With \( C = 3 \) and by the Lemma, \( \text{size}(t) \leq 3(2^d - 1) < C \cdot 2^d \).
7. (20 points) The *Theorem on Loop Invariants* from Chapter 5 says that to prove an assertion \textsc{post} holds after the while-loop executes,

\[
\begin{array}{c}
\text{while test do } \{ \text{inv} \} \text{ body} \\
\{ \text{post} \}
\end{array}
\]

it suffices to prove:

\[
\begin{array}{c}
\text{initialization: } \text{pre} \implies \text{inv}. \\
\text{invariance: } \{ \text{inv} \land \text{test} \} \text{ body} \{ \text{inv} \} \\
\text{termination: } \text{inv} \land \neg \text{test} \implies \text{post}.
\end{array}
\]

Use the Theorem on Loop Invariants to prove the program below computes $A^B$.

\[
\begin{array}{c}
\{ x = A \land y = B \} \\
\begin{array}{c}
\text{begin} \\
z := 1; \\
\text{while } y \neq 0 \text{ do } \{ z \cdot x^y = A^B \} \\
\quad \text{begin} \\
\quad \text{while even?}(y) \text{ do } \{ z \cdot x^y = A^B \land y \neq 0 \} \\
\qquad \text{begin} y := 1/2y; \quad x := x \times x \quad \text{end}; \\
\qquad z := z \times x; \\
\quad y := y - 1 \\
\quad \text{end} \\
\text{end} \\
\{ z = A^B \}
\end{array}
\end{array}
\]

There are two loops in the program, so there are two initialization, invariance and termination arguments.

**(INNER LOOP)**

**initialization.** The each time the program reaches the inner loop, the outer loop’s invariant is true, and this, together with the loop test even?($y$) is just the inner loop’s invariant.

**invariance.** If $y$ is even, the inner loop body computes new values, $x' = x^2$, $y' = 1/2y \neq 0$ and $z' = z$. So $z' \cdot x^{y'} = z \cdot (x^2)^{1/2} = z \cdot x^y = A^B$. So the inner invariant is preserved.

**termination.** When the inner loop terminates, its invariant is true and $y$ is no longer an even number. Hence it is also the case that the outer invariant remains true.

**(OUTER LOOP)**

**initialization.** When the program first reaches the outer loop, we have $x = A$, $y = B$ and $z = 1$, so $z \cdot x^y = 1 \cdot A^B = A^B$.

**invariance.** The outer invariant still holds after the inner loop. The body of the outer loop assigns new values $x' = x$, $y' = y - 1$ and $z' = zx$. Thus, $z' \cdot x^{y'} = z \cdot x^{y-1} = z \cdot x^{(y-1)} = z \cdot x^y = A^B$ and the invariant is preserved.

**termination.** On termination we still have $z \cdot x^y = A^B$, but $y = 0$, so $x^y = 1$. Therefore the condition $z = A^B$ holds as desired.
8. (12 points) Let $A$ be a set and $R \subseteq A \times A$ a relation. On the left below is an inductive definition of a relation $R^* \subseteq A \times A$. The statements on the right explain what this inductive definition means. They define a sequence of relations, $R_i \subseteq A \times A$, whose union (or limit) is $R^*$.

1. $R \subseteq R^*$
2. $(a, b), (b, c) \in R^* \Rightarrow (a, c) \in R^*$
3. nothing else

1. $R_0 = R$
2. $R_{k+1} = \bigcup_{i=0}^{\infty} \{(a, c) \mid \exists b \in A: (a, b), (b, c) \in R_k\}$
3. $R^* = \bigcup_{i=0}^{\infty} R_i$

(a) For the relation depicted below-right, what are the sets $R_0$, $R_1$, $R_2$, $R_3$?

$$R_0 = \{(a, b), (b, d), (b, c), (e, f), (f, e)\}$$

$$R_1 = \{(a, b), (b, d), (b, c), (e, f), (f, e), (a, d), (a, c), (f, f), (e, e)\}$$

$$R_2 = (\text{same as } R_1)$$

$$R_3 = (\text{same as } R_2)$$

(b) The “nothing else” clause means that there are no unnecessary ordered pairs in $R^*$. **Prove:** If $S$ is any transitive relation that contains $R$, then $R^* \subseteq S$. **HINT:** use the more primitive definition of $R^* = \bigcup_{i=0}^{\infty} R_i$.

**Proposition.** For all $n \in \mathbb{N}$, $R_n \subseteq S$.

**Proof.** The proof is by induction on $n$. In the base case, it is given that $R_0 \subseteq S$.

(INDUCTION) suppose $R_k \subseteq S$ . . . and let $(x, y)$ be any element of $R_{k+1}$ . . . Therefore $R_{k+1} \subseteq A$.

Now let $(x, y)$ be any element of $R^*$. Then because $R^* = \bigcup R_k$, there must be some $k$ for which $(x, y) \in R_k$. By the proposition, $R_k \subseteq S$, so $(x, y) \in S$. Since $(x, y)$ was arbitrary, we have shown that $R^* \subseteq S$. 

\[ \square \]
9. (10 points)

(a) Draw an automaton over \( A = \{b, i, p, t, y, o\} \) that accepts the language \{bippity, boppity, boo\}. You may use empty and nondeterministic transitions if you wish.

\[ \text{One that works is} \]

(b) Draw an automaton over \( \{0, 1\} \) that accepts a language whose words all start and end with the same two letters, such as 10110 and 000, but not 1001. You may use empty and nondeterministic transitions if you wish.

\[ \text{One that works is} \]