• You have **50 minutes** to complete this exam.

• This is a **closed-book** exam.

• Do not share anything with other students. Do not talk to other students. Do not look other students’ exams. Do not expose your exam to easy viewing by other students. Violation of any of these rules will be considered cheating.

• If you believe there is an error or an ambiguous question, you may seek clarification from the instructor. Please speak quietly or write your question out.

• Including this cover sheet and rules at the end, there are 11 pages to the exam, including one blank page for workspace. Once the exam begins (and not before!), please verify that you have all 11 pages.

• Please write your name and NetID in the spaces above, and also in the provided space at the top of every sheet.

• Show your work. Partial credit will be given for incomplete answers.

• When writing proof trees, remember that you can abbreviate expressions, states, etc. as long as you make it clear what the abbreviations mean.

• The pages at the end of the exam contain inference rules for various systems. You may detach these pages. If you do, please turn them in with the rest of your exam.

• If you finish with time remaining, check your work!
<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1. (27 points)
The semantic rules for a simple object-oriented language are given in Appendix A. Suppose the following classes are declared in the class table CT:

```java
class Computer extends Object{
    int memUsed;
    bool on;

    Computer(int memUsed, bool on){
        super();
        this.memUsed = memUsed;
        this.on = on;
    }

    bool bootUp(){
        this.on = true;
        this.OSVersion = this.OSVersion + 1;
        this.memUsed = 0;
        return true;
    }

    int shutDown(){
        this.on = false;
        return this.memUsed;
    }
}

class Mac extends Computer{
    int OSVersion;

    Mac(int memUsed, bool on, int OSVersion){
        super(memUsed, on);
        this.OSVersion = OSVersion;
    }

    int updateOS(){
        this.OSVersion = this.OSVersion + 1;
        return true;
    }
}
```

(a) (8 points) Write the next configuration that

```
(myComp.memUsed = backup.memUsed;, nil, {myComp = r1, backup = r2}, {r1 -> new Mac(100, true, 10), r2 -> new Computer(50, false)})
```

steps to.

**Solution:** (skip, nil, {myComp = r1, backup = r2}, {r1 -> new Mac(50, true, 10), r2 -> new Computer(50, false)})
(b) (19 points) Write a proof tree for the judgment

\[(\text{myComp.memUsed = backup.memUsed; nil, \{myComp = r}_1, \text{backup = r}_2\}, \{r}_1 \mapsto \text{new Mac(100, true, 10), r}_2 \mapsto \text{new Computer(50, false)}\}) \rightarrow g\]

where \(g\) is your answer to part a.

**Solution:**

Let \(\rho_1 \) be \(\{\text{myComp = r}_1, \text{backup = r}_2\}\) and \(\sigma_1 \) be \(\{r}_1 \mapsto \text{new Mac(50, true, 10), r}_2 \mapsto \text{new Computer(50, false)}\)\).

\[
\begin{array}{c}
\text{myComp, } \rho_1, \sigma_1 \downarrow r_1 \\
\text{σ}_1(r_1) = \text{new Mac(50, true, 10)} \\
\text{backup, } \rho_1, \sigma_1 \downarrow r_2 \\
\sigma_1(r_2) = \text{new Computer(50, false)} \\
\text{backup.memUsed, } \rho_1, \sigma_1 \downarrow 50 \\
\text{myComp.memUsed = backup.memUsed; nil, \{myComp = r}_1, \text{backup = r}_2\}, \{r}_1 \mapsto \text{new Mac(100, true, 10), r}_2 \mapsto \text{new Computer(50, false)}\}) \rightarrow \\
\text{skip, nil, \{myComp = r}_1, \text{backup = r}_2\}, \{r}_1 \mapsto \text{new Mac(50, true, 10), r}_2 \mapsto \text{new Computer(50, false)}\})
\end{array}
\]
Problem 2. (26 points)

(a) (10 points) In each of the following terms, rename variables so that no two distinct variables have the same name, without changing the meaning of the terms.

- \((\lambda x. (x y)) (\lambda z. (z x))\)

  **Solution:** \((\lambda w. (w y)) (\lambda z. (z x))\)
  
  Be careful not to rename free variables!

- \(\lambda x. \lambda y. \lambda y. ((y y) x)\)

  **Solution:** \(\lambda x. \lambda y. \lambda z. ((z z) x)\)

- \((\lambda x. ((\lambda x. x) x)) (\lambda x. x)\)

  **Solution:** \((\lambda x. ((\lambda y. y) x)) (\lambda z. z)\)

(b) (16 points) Evaluate each of the following terms according to call-by-value semantics for as many steps as possible. Show each step.

- \((\lambda x. (x y)) (\lambda z. (z x))\)

  **Solution:** \((\lambda x. (x y)) (\lambda z. (z x)) \rightarrow (\lambda z. (z x)) y \rightarrow y x\)
• (The function that takes an argument and applies that argument to itself) applied to (the function that takes an argument and returns its argument).

Solution: \((\lambda x. (x \ x)) (\lambda x. x) \rightarrow (\lambda x. x) (\lambda x. x) \rightarrow (\lambda x. x)\)

• \(((\lambda x. \lambda y. x) \ y) ((\lambda x. x) \ z)\)

Solution: \(((\lambda x. \lambda y. x) \ y) ((\lambda x. x) \ z) \rightarrow (\lambda w. y) ((\lambda x. x) \ z) \rightarrow (\lambda w. y) z \rightarrow y\)
Problem 3. (18 points)
The typing rules for the simply-typed lambda calculus are given in Appendix [3]. Write the type of each of the following terms.

- \( \lambda x : \text{int} \cdot (3 + x) \)

**Solution:** int \( \rightarrow \) int

This is a function that takes an int argument and returns an int.

- \( (\lambda x : \text{int} \cdot (x + x)) \ ((\lambda y : \text{int} \cdot y) \ 5) \)

**Solution:** int

The LHS is a function of type int \( \rightarrow \) int. The RHS has a function of type int \( \rightarrow \) int applied to an argument of type int, so its overall type is int. So the whole expression is a function of type int \( \rightarrow \) int applied to an argument of type int, yielding an int.

- \( \lambda f : \text{int} \rightarrow \text{int} \rightarrow \text{int} \cdot \lambda x : \text{int} \cdot (f \ x) \)

**Solution:** (int \( \rightarrow \) int \( \rightarrow \) int) \( \rightarrow \) int \( \rightarrow \) int or, equivalently, (int \( \rightarrow \) int \( \rightarrow \) int) \( \rightarrow \) int \( \rightarrow \) (int \( \rightarrow \) int)

This is a function that takes two arguments, \( f \) of type int \( \rightarrow \) int \( \rightarrow \) int and \( x \) of type int. Its return type is the type of \( f \ x \). Given the types of \( f \) and \( x \), we know that \( f \ x \) is of type int \( \rightarrow \) int, so the overall function takes an (int \( \rightarrow \) int \( \rightarrow \) int) and an int and returns an (int \( \rightarrow \) int).
Problem 4. (29 points)

The semantic rules for a simple functional language are given in Appendix C.

(a) (10 points) What does the program \(f\ x\) evaluate to when run in the environment where 
\(f = \langle \text{fun } y \to \text{match } y \text{ with } \text{inl } a \to a \mid \text{inr } b \to b + x, \{x = 2\} \rangle\) and \(x = \text{inr } 5\)?

Solution: 7

(b) (19 points) Write a proof tree for the judgment

\[
(f, \{x = \text{inr } 5, f = \langle \text{fun } y \to \text{match } y \text{ with } \text{inl } a \to a \mid \text{inr } b \to b + x, \{x = 2\} \rangle \} ) \Downarrow v
\]

where \(v\) is your answer to part a.

Solution:
Let \(\rho_2\) be \(\{x = \text{inr } 5, f = \langle \text{fun } y \to \text{match } y \text{ with } \text{inl } a \to a \mid \text{inr } b \to b + x, \{x = 2\} \rangle \}, \rho_3\) be \(\{x = 2, y = \text{inr } 5, b = 5\}\), and \(m\) be \(\text{match } y \text{ with } \text{inl } a \to a \mid \text{inr } b \to b + x\).

\[
\frac{(f, \rho_2) \Downarrow \langle \text{fun } y \to m, \{x = 2\} \rangle \qquad (x, \rho_2) \Downarrow \text{inr } 5 \qquad (y, \{x = 2, y = \text{inr } 5\}) \Downarrow \text{inr } 5 \qquad (b + x, \rho_3) \Downarrow 7}{(m, \{x = 2, y = \text{inr } 5\}), \Downarrow 7} \]

\[
\frac{(f, \rho_2) \Downarrow \langle \text{fun } y \to m, \{x = 2\} \rangle \qquad (x, \rho_2) \Downarrow \text{inr } 5 \qquad (m, \{x = 2, y = \text{inr } 5\}), \Downarrow 7}{(f x, \{x = \text{inr } 5, f = \langle \text{fun } y \to \text{match } y \text{ with } \text{inl } a \to a \mid \text{inr } b \to b + x, \{x = 2\} \rangle \} ) \Downarrow 7}
\]
A  Operational Semantics for a Simple Object-Oriented Language

\( n \) is an integer literal \((n, \rho, \sigma) \Downarrow n\)

\( b \) is a boolean literal \((b, \rho, \sigma) \Downarrow b\)

\( x, \rho, \sigma \Downarrow v \)

\( (\rho(x) = v) \Downarrow v_1 \)

\( (e_1, \rho, \sigma) \Downarrow v_1 \)

\( (e_2, \rho, \sigma) \Downarrow v_2 \)

\( v_1 \oplus v_2 = v \)

where \( \oplus \) is an arithmetic operator

\( e, \rho, \sigma \Downarrow v \)

\( \sigma(r) = \text{new } C(v_1, ..., v_n) \) \( (\text{fields}(C)[i] = \tau f) \)

\( (c_1, k, \rho, \sigma) \rightarrow (c_1', k, \rho, \sigma) \)

\( (e_1, \rho, \sigma) \Downarrow v_1 \) \( (e_n, \rho, \sigma) \Downarrow v_n \) \( (r \notin \text{dom}(\sigma)) \)

\( x = \text{new } C(e_1, ..., e_n); k, \rho, \sigma \rightarrow (\text{skip}, k, \rho[x \mapsto r], \sigma[r \mapsto \text{new } C(v_1, ..., v_n)]) \)

\( e, \rho, \sigma \Downarrow v \)

\( \sigma(r) = \text{new } C(v_1, ..., v_n) \) \( (\text{fields}(C)[i] = \tau f) \)

\( (e_1, \rho, \sigma) \Downarrow v_1 \) \( (e_n, \rho, \sigma) \Downarrow v_n \)

\( (x = e.m(e_1, ..., e_n); k, \rho, \sigma) \rightarrow (e, ((\sigma, x) :: k), \{\text{this } = r, x_1 = v_1, ..., x_n = v_n\}, \sigma) \)

\( \text{return } e; ((\rho_0, x) :: k), \rho, \sigma \rightarrow (\text{skip}, k, \rho_0[x \mapsto v], \sigma) \)

B  Typing Rules for Simply-Typed Lambda Calculus

\( n \) is an integer literal \( \Gamma \vdash n : \text{int} \)

\( \Gamma \vdash e_1 + e_2 : \text{int} \)

\( \Gamma \vdash \text{int} \rightarrow \text{int} \rightarrow \text{int} \)

\( \Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \)

\( \Gamma \vdash e_2 : \tau_1 \rightarrow \tau_2 \)

\( \Gamma \vdash (\lambda x : \tau_1. e) : \tau_1 \rightarrow \tau_2 \)

\( \Gamma \vdash (e_1 e_2) : \tau_2 \)
C  Operational Semantics for a Simple Functional Language

\[
\begin{align*}
(n, \rho) &\Downarrow n & (b, \rho) &\Downarrow b \\
\frac{}{(e_1, \rho) \Downarrow v_1}{(e_2, \rho) \Downarrow v_2}{(v_1 \oplus v_2 = v)}{(e_1 \oplus e_2, \rho) \Downarrow v} \\
\frac{}{(e_1, \rho) \Downarrow (\text{fun } x -> e, \rho')}{(e_2, \rho) \Downarrow v_2}{(e, \rho'[x \mapsto v_2]) \Downarrow v}{(e_1 e_2, \rho) \Downarrow v} \\
\frac{}{(e, \rho) \Downarrow \text{inl } v}{(e_1, \rho[x_1 \mapsto v]) \Downarrow v'}{(\text{match } e \text{ with inl } x_1 \rightarrow e_1 \mid \text{inr } x_2 \rightarrow e_2, \rho) \Downarrow v'}
\end{align*}
\]
D  Scratch Space