Before starting, be sure that you understand the course policy on Academic Integrity.

1. **PM: A Pattern Matching Language**

Recall the pattern matching language from Homework 03.

**Syntax definition:**

\[
\begin{align*}
v & ::= c \mid (v, v) \mid s(v) \\
p & ::= x \mid c \mid (p, p) \mid s(p) \mid \ldots (p) \\
b & ::= \_ \mid b, x \mapsto v \\
(c & \in \{\ldots, -2, -1, 0, 1, 2, \ldots\}) \\
(s & \text{ any non-empty string of letters}) \\
x & \text{ any non-empty string of letters}
\end{align*}
\]

**Formal large-step operational semantics**

\[
\begin{align*}
p;v \Downarrow b \\
L\text{Wild} & \quad L\text{Var} & \quad L\text{Const} \\
\_;v & \Downarrow \_ & x;v \Downarrow x \mapsto v & \quad c;c \Downarrow c \\
L\text{Pair} & \quad L\text{Tag} & \quad L\text{DescPair1} & \quad L\text{DescPair2} \\
p_1;v_1 \Downarrow b_1 & \quad p_2;v_2 \Downarrow b_2 & \quad p_s;v_s \Downarrow b \\
(p_1,p_2);(v_1,v_2) \Downarrow b_1 \@ b_2 & \quad s(p_s);s(v_s) \Downarrow b \\
L\text{DescTag} & \quad L\text{DescVal} \\
\ldots (p_d);v_1 \Downarrow b & \quad \ldots (p_d);v_2 \Downarrow b & \quad \ldots (p_d);v_s \Downarrow b & \quad \ldots (p_d);v \Downarrow b \\
\ldots (p_d);(v_1,v_2) \Downarrow b & \quad \ldots (p_d);(v_1,v_2) \Downarrow b & \quad \ldots (p_d);s(v_s) \Downarrow b & \quad \ldots (p_d);v \Downarrow b
\end{align*}
\]
Formal small-step operational semantics

\[
p;v;b \rightarrow p';v';b'
\]

\[
\begin{array}{ccc}
\text{SVar} & \quad & \text{SConst} \\
\frac{x;v;b \rightarrow _\tau w;b, x \mapsto v}{x;v;b \rightarrow _\tau w;b}
\end{array}
\]

\[
\begin{array}{ccc}
\text{SPair1} & \quad & \text{SPair2} & \quad & \text{STag} \\
\frac{p_1;v_1;b \rightarrow p'_1;v'_1;b'}{(p_1,p_2);(v_1,v_2);b \rightarrow (p'_1,p_2);(v'_1,v_2);b'} & \frac{\langle_\tau ,p_2\rangle;(v_1,v_2);b \rightarrow p_2;v_2;b}{\langle_\tau ,p_2\rangle;(v_1,v_2);b \rightarrow p_2;v_2;b}
\end{array}
\]

\[
\begin{array}{ccc}
\text{SDescPair1} & \quad & \text{SDescPair2} \\
\frac{\ldots (p_d);(v_1,v_2);b \rightarrow \ldots (p_d);v_1;b}{\ldots (p_d);(v_1,v_2);b \rightarrow \ldots (p_d);v_1;b}
\end{array}
\]

\[
\begin{array}{ccc}
\text{SDescTag} & \quad & \text{SDescVal} \\
\frac{\ldots (p_d);s(v_s);b \rightarrow \ldots (p_d);v_s;b}{\ldots (p_d);v_s;b \rightarrow \ldots (p_d);v_s;b}
\end{array}
\]

Formal small-steps operational semantics

\[
p;v;b \rightarrow^* p';v';b'
\]

\[
\begin{array}{ccc}
\text{S* Zero} & \quad & \text{S* Step} \\
\frac{p;v;b \rightarrow^* p;v;b}{p;v;b \rightarrow^* p;v;b}
\end{array}
\]

\[
\begin{array}{ccc}
\frac{p;v;b \rightarrow p_1;v_1;\vdash}{p;v;b \rightarrow p_1;v_1;\vdash} & \frac{p_1;v_1;\vdash \rightarrow p_2;v_2;\vdash}{p_1;v_1;\vdash \rightarrow p_2;v_2;\vdash}
\end{array}
\]
Choose either (a) or (b) to complete and submit.

(a) Prove the following:

i. Lemma (small-step match prefix large-step match)
   If \( p;v; b_a \rightarrow^* p'; v'; b'_a \) and \( p'; v' \downarrow b'_b \), then there exists \( b_b \) such that \( p; v \downarrow b_b \).

   **Hints:**
   - The proof structure is similar to a proof we sketched in Lecture 06.
   - Use induction on the assumed derivation.
   - The proof will have cases for each rule in the small-step semantics and you must show that a derivation exists in the large-step semantics.
   - Note that the lemma does not state that there is any relationship between \( b_b \) and \( b_a \) or \( b'_b \). Doing so appears to be much more difficult.
   - The proof of this lemma is longer than the proof of the following theorem.

ii. Theorem (large-step match implies small-steps match)
   If \( p;v; b_a \rightarrow^* \perp; v'; b'_a \), then there exists \( b_b \) such that \( p; v \downarrow b_b \).

   **Hints:**
   - The proof structure is similar to a proof we sketched in Lecture 06.
   - Use induction on the assumed derivation.
   - The proof will have cases for each rule in the small-steps semantics and you must show that a derivation exists in the large-step semantics.
   - Use Lemma (small-step match prefix large-step match).
   - Note that the theorem does not state that there is any relationship between \( b_b \) and \( b_a \) or \( b'_b \). Doing so appears to be much more difficult.
   - The proof of this theorem is shorter than the proof of the preceding lemma.

(b) Prove the following:

i. Lemma (small-steps lift through pair)
   If \( p_a; v_a; b \rightarrow^* p'_a; v'_a; b' \), then \( (p_a, p_b) ; (v_a, v_b) ; b \rightarrow^* (p'_a, p_b) ; (v'_a, v_b) ; b' \).

   **Hints:**
   - Use induction on the assumed derivation.
   - The proof will have cases for each rule in the small-steps semantics and you must show that a derivation exists in the small-steps semantics.
   - The proof of this lemma is shorter than the proof of the following theorem.

ii. Theorem (large-step match implies small-steps match)
   If \( p; v \downarrow b \), then for all \( b \), there exists \( v' \) such that \( p; v; b \rightarrow^* \perp; v'; b @ b \).

   **Hints:**
   - The proof structure is similar to a proof we sketched in Lecture 06.
   - Use induction on the assumed derivation.
   - The proof will have cases for each rule in the large-step semantics and you must show that a derivation exists in the small-steps semantics.
   - Use Lemma (small-steps lift through pair).
   - Use, without proof, Lemma (small-steps transitive): If \( p_a; v_a; b_a \rightarrow^* p_b; v_b; b_b \) and \( p_b; v_b; b_b \rightarrow^* p_c; v_c; b_c \), then \( p_a; v_a; b_a \rightarrow^* p_c; v_c; b_c \).
   - Note that the theorem uses \( b \) twice; it states that the small-step semantics produces the same binding list (appended to \( b \)). You may assume, without proof, that \( b \rightarrow^* \perp \), that \( b \rightarrow \perp, x \rightarrow v = b, x \rightarrow v \), and that \( (b \otimes (b_1) @ b_2 = b \otimes (b_1 @ b_2) \).
   - Note that the theorem is stronger than what we actually want (which is that \( p; v; \rightarrow^* \perp; v'; b \)). You will need the stronger claim in one case of the proof.
   - The proof of this theorem is longer than the proof of the preceding lemma.
2. \(\lambda\)-calculus

Recall the encodings of booleans and natural numbers in the Lambda Calculus:

\[
\begin{align*}
\text{true} &= \lambda x. \lambda y. x \\
\text{false} &= \lambda x. \lambda y. y \\
\text{if} &= \lambda b. \lambda t. \lambda f. b \ t \ f \\
0 &= \lambda s. \lambda z. z \\
1 &= \lambda s. \lambda z. s \ z \\
2 &= \lambda s. \lambda z. s \ (s \ z) \\
3 &= \lambda s. \lambda z. s \ (s \ (s \ z)) \\
\text{succ} &= \lambda n. \lambda s. \lambda z. s \ (n \ s \ z) \\
\text{plus} &= \lambda m. \lambda n. m \ \text{succ} \ n \\
\text{times} &= \lambda m. \lambda n. m \ (\text{plus} \ n) \ 0 \\
\text{isZero} &= \lambda n. n \ (\lambda x. \text{false}) \ \text{true}
\end{align*}
\]

Note that all of these abbreviations are values.

(a) Show how to write a term \text{and} that performs the logical conjunction of two booleans.

(b) Show the small-step, left-to-right, call-by-value (CBV) reduction of the term \text{and \ true \ false}. Expand abbreviations as necessary so that every reduction step shows the explicit \(\lambda x. e\) term that is being reduced.

(c) Show how to write a term \text{or} that performs the logical disjunction of two booleans.

(d) Show the small-step, left-to-right, call-by-value (CBV) reduction of the term \text{or \ true \ false}.

(e) Show how to write a term \text{not} that performs the logical negation of a boolean.

(f) Show how to write a term \text{isEven} that determines whether or not a number is even. It should return \text{true} when the number is even and \text{false} otherwise.

(g) Show the small-step, left-to-right, call-by-value (CBV) reduction of the term \text{isEven (succ 1)}. 
Debriefing

- How many hours did you spend on this assignment?
- Would you rate it as easy, moderate, or difficult?
- How deeply do you feel you understand the material it covers (0% – 100%)?
- If you have any other comments about the assignment, then please include them with your submission or send email to mtf@cs.rit.edu.

Submission

All components of the assignment:

- problems (1a or 1b), 2a, 2b, 2c, 2d, 2e, 2f, 2g, and debriefing in a file named homework04.pdf

must be submitted to the Homework 4 Assignment on MyCourses by the due date.

For handwritten components, please use either a document scanner or a mobile scanning app like Adobe Scan or Microsoft Lens.