1 Warm-up (10%)

1. (a) let x be 3 in
   let f be (fun g(y:Int):Int is x+y) in
   let x be 4 in
   f(x)

(b) let f be (fun g(y:Int):Int is 3+y) in
    let x be 4 in
    f(x)

(c) let x be 4 in
    (fun g(y:Int):Int is 3+y)(x)

(d) (fun g(y:Int):Int is 3+y)(4)

(e) 3+4

(f) 7

2. (a) let fact be (fun g(y:Int):Int is if y<1 then 1 else y*g(y-1)) in
      fact 2

(b) (fun g(y:Int):Int is if y<1 then 1 else y*g(y-1))(2)

(c) if 2<1 then 1
    else 2*(fun g(y:Int):Int is if y<1 then 1 else y*g(y-1))(2 - 1)

(d) 2*(fun g(y:Int):Int is if y<1 then 1 else y*g(y-1))(2 - 1)

(e) 2*(fun g(y:Int):Int is if y<1 then 1 else y*g(y-1))(1)
2 Adding pairs to NQSML. (65%)

Consider the following extension of NQSML of adding pairs. The abstract syntax is extended as follows:

```
v ::= ···
    | ⟨v₁, v₂⟩

e ::= ···
    | ⟨e₁, e₂⟩
    | e₁
    | e₂

t, u ::= ···
    | t₁ × t₂
```

A pair of values is considered a value. The new syntax allows creation of a pair, and operations to project out the first or second component of a pair. Unlike SML, there is no pattern-matching for pairs, so in this abstract syntax a function to raise an integer to a positive integer power would look like:

```
fun p(arg:Int×Int):Int is
    let x be arg.1 in
    let n be arg.2 in
    if (n < T) then T else p(⟨x, n − T⟩)
```

The new typing rules are:

\[
\Gamma ⊢ e₁ : t₁ \quad \Gamma ⊢ e₂ : t₂
\]

\[
\Gamma ⊢ ⟨e₁, e₂⟩ : t₁ × t₂
\]  (26)
\[\Gamma \vdash e : t_1 \times t_2 \quad \Gamma \vdash e.1 : t_1 \quad \Gamma \vdash e : t_1 \times t_2 \quad \Gamma \vdash e.2 : t_2 \] (27) (28)

and the new evaluation rules are:

\[
\begin{align*}
    e_1 & \rightarrow e'_1 \\
    \langle e_1, e_2 \rangle & \rightarrow \langle e'_1, e_2 \rangle \\
    e_2 & \rightarrow e'_2 \\
    \langle v_1, e_2 \rangle & \rightarrow \langle v_1, e'_2 \rangle \\
    e & \rightarrow e' \\
    e.1 & \rightarrow e'.1 \\
    \langle v_1, v_2 \rangle.1 & \rightarrow v_1 \\
    e & \rightarrow e' \\
    e.2 & \rightarrow e'.2 \\
    \langle v_1, v_2 \rangle.2 & \rightarrow v_2
\end{align*}
\] (29) (30) (31) (32) (33) (34)

Lemma 1 (Inversion Extension)
1. If \( \Gamma \vdash \langle e_1, e_2 \rangle : t \) then \( t = t_1 \times t_2 \) for some types \( t_1 \) and \( t_2 \), where \( \Gamma \vdash e_1 : t_1 \) and \( \Gamma \vdash e_2 : t_2 \).

2. If \( \Gamma \vdash e.1 : t_1 \) then \( \Gamma \vdash e : t_1 \times t_2 \) for some type \( t_2 \).

3. If \( \Gamma \vdash e.2 : t_2 \) then \( \Gamma \vdash e : t_1 \times t_2 \) for some type \( t_1 \).

Lemma 2 (Canonical Forms Extension)
1. If \( \vdash v : t_1 \times t_2 \) then \( v \) is an pair of the form \( \langle v_1, v_2 \rangle \).

Proposition 3 (Type Preservation)
If \( \vdash e : t \) and \( e \rightarrow e' \) then \( \vdash e' : t \).

Proof: By induction on the proof that \( e \rightarrow e' \).

- Case: Rule 29. Then \( e = \langle e_1, e_2 \rangle \) and \( t = t_1 \times t_2 \). By Inversion, \( \vdash e_1 : t_1 \) and \( \vdash e_2 : t_2 \). By the inductive hypothesis applied to \( e_1 \rightarrow e'_1 \), we have \( \vdash e'_1 : t_1 \). Therefore \( \vdash \langle e'_1, e_2 \rangle : t_1 \times t_2 \) as required.

- Case: Rule 30. Then \( e = \langle v_1, e_2 \rangle \) and \( t = t_1 \times t_2 \). By Inversion, \( \vdash v_1 : t_1 \) and \( \vdash e_2 : t_2 \). By the inductive hypothesis applied to \( e_2 \rightarrow e'_2 \), we have \( \vdash e'_2 : t_2 \). Therefore \( \vdash \langle v_1, e'_2 \rangle : t_1 \times t_2 \) as required.
• Case: Rule 31. Then \( e = e_1 .1 \) and \( e' = e'_1 .1 \) where \( e_1 \rightarrow e'_1 \). By inversion, \( \vdash e_1 : t \times t_2 \) for some type \( t_2 \). By the inductive hypothesis, \( \vdash e'_1 : t \times t_2 \). Thus \( \vdash e'_1 .1 : t \) as required.

• Case: Rule 32. Then \( e = \langle v_1, v_2 \rangle .1 \) and \( e' = v_1 \). By inversion, \( \vdash \langle v_1, v_2 \rangle : t \times t_2 \) for some type \( t_2 \). By inversion again, \( \vdash v_1 : t \), as required.

• Case: Rule 33 and Rule 34. Exactly analogous to the two previous cases.

Proposition 4 (Progress)
If \( \vdash e : t \) then either \( e \) is a value or there exists \( e' \) such that \( e \rightarrow e' \).

Proof: By induction on the proof of \( \vdash e : t \), and cases on the last rule used.

• Case: Rule 26. Then \( e = \langle e_1, e_2 \rangle \) and \( t = t_1 \times t_2 \) and there are sub-proofs \( \vdash e_1 : t_1 \) and \( \vdash e_2 : t_2 \). There are three subcases to consider.
  - Subcase: \( e_1 \) is not a value. By the inductive hypothesis, there exists \( e'_1 \) such that \( e_1 \rightarrow e'_1 \). Thus by Rule 29 we have \( \vdash \langle e_1, e_2 \rangle \rightarrow \langle e'_1, e_2 \rangle \).
  - Subcase: \( e_1 \) is a value, but \( e_2 \) is not. By the inductive hypothesis there exists \( e'_2 \) such that \( e_2 \rightarrow e'_2 \). By Rule 30 we have \( \vdash \langle e_1, e_2 \rangle \rightarrow \langle e_1, e'_2 \rangle \).
  - Subcase: \( e_1 \) and \( e_2 \) are both values. Then the pair \( \langle e_1, e_2 \rangle \) is also a value.

• Case: Rule 27. Then \( e = e_1 .1 \) and and there is a sub-proof \( \vdash e_1 : t \times t_2 \) for some type \( t_2 \). There are two subcases to consider:
  - Subcase: \( e_1 \) is not a value. By the inductive hypothesis there exists \( e'_1 \) such that \( e_1 \rightarrow e'_1 \). Thus \( e \rightarrow e'_1 .1 \).
  - Subcase: \( e_1 \) is a value. By the Canonical Forms lemma, \( e_1 \) must be of the form \( \langle v_1, v_2 \rangle \) for some values \( v_1 \) and \( v_2 \). Thus by Rule 32 we have \( e \rightarrow v_1 \).

• Case: Rule 28. Exactly analogous to the previous case.

3 Natural Semantics (25%)

\[
\frac{v \Downarrow v}{v} \quad (35)
\]
\[
\frac{e_1 \Downarrow m_1 \quad e_2 \Downarrow m_2}{e_1 + e_2 \Downarrow m_1 + m_2} \quad (36)
\]
\[
\frac{e_1 \Downarrow m_1 \quad e_2 \Downarrow m_2}{e_1 < e_2 \Downarrow m_1 < m_2} \quad (37)
\]
1. Proposition 5

If \( e \downarrow v \) then \( e \rightarrow^* v \).

Proof: By induction on the proof of \( e \downarrow v \), and cases on the last rule used.

- Case: Rule 35. Then \( v \rightarrow^* v \) because \( \rightarrow^* \) is reflexive.

- Case: Rule 36. Then \( e = e_1 + e_2 \) and \( v = m_1 + m_2 \), and there are subproofs of \( e_1 \downarrow m_1 \) and \( e_2 \downarrow m_2 \). Then by the inductive hypothesis we have \( e_1 \rightarrow^* m_1 \) and \( e_2 \rightarrow^* m_2 \). Therefore \( e_1 + e_2 \rightarrow^* m_1 + e_2 \) and \( m_1 + e_2 \rightarrow^* m_1 + m_2 \) and \( m_1 + m_2 \rightarrow m_1 + m_2 \). By transitivity, \( e_1 + e_2 \rightarrow^* m_1 + m_2 \).

- Case: Rule 37. Analogous to previous case.

- Case: Rule 38. Then \( e = \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \) and there are subproofs of \( e_1 \downarrow \top \) and \( e_2 \downarrow v \). By the inductive hypothesis, \( e_1 \rightarrow^* \top \) and \( e_2 \rightarrow^* v \). Thus

\[
\begin{align*}
\text{if } e_1 \downarrow \top \text{ and } e_2 \downarrow v \\
\text{then } e_1 \rightarrow^* \top \text{ and } e_2 \rightarrow^* v \\
\text{and } e_1 \rightarrow^* \top \text{ and } e_2 \rightarrow^* v \\
\text{and } e_1 \rightarrow^* \top \text{ and } e_2 \rightarrow^* v.
\end{align*}
\]

- Case: Rule 39. Analogous to previous case.

- Case: Rule 40. Then \( e = \text{let } x \text{ be } e_1 \text{ in } e_2 \) and there are subproofs of \( e_1 \downarrow v_1 \) and that \( e_2[x \mapsto v_1] \downarrow v \). By the inductive hypothesis, \( e_1 \rightarrow^* v_1 \). Thus

\[
\begin{align*}
\text{let } x \text{ be } e_1 \text{ in } e_2 \\
\rightarrow^* \text{let } x \text{ be } v_1 \text{ in } e_2 \\
\text{and } e_2[x \mapsto v_1] \rightarrow^* v \\
\text{and } e_2[x \mapsto v_1] \rightarrow^* v \\
\text{and } e_2[x \mapsto v_1] \rightarrow^* v.
\end{align*}
\]

- Case: Rule 41. Then \( e = e_1 e_2 \) and there are subproofs of \( e_1 \downarrow (\text{fun } f(x:t_1):t_2 \text{ is } e'_1) \) and \( e_2 \downarrow v_2 \) and

\[
\begin{align*}
\text{let } x \text{ be } e_1 \text{ in } e_2 \\
\rightarrow^* \text{let } x \text{ be } e_1 \text{ in } e_2 \\
\text{and } \text{fun } f(x:t_1):t_2 \text{ is } e'_1 \text{ and } e_2 \rightarrow^* v_2 \text{ and } e \rightarrow^* v \\
\rightarrow^* \text{fun } f(x:t_1):t_2 \text{ is } e'_1 \text{ and } e_2 \rightarrow^* v_2 \text{ and } e \rightarrow^* v.
\end{align*}
\]

Therefore

\[
\begin{align*}
\text{let } x \text{ be } e_1 \text{ in } e_2 \\
\rightarrow^* \text{let } x \text{ be } e_1 \text{ in } e_2 \\
\text{and } \text{fun } f(x:t_1):t_2 \text{ is } e'_1 \text{ and } e_2 \rightarrow^* v_2 \text{ and } e \rightarrow^* v \\
\rightarrow^* \text{fun } f(x:t_1):t_2 \text{ is } e'_1 \text{ and } e_2 \rightarrow^* v_2 \text{ and } e \rightarrow^* v.
\end{align*}
\]

as required.
2.

**Proposition 6**

If \( e \rightarrow e' \) and \( e' \Downarrow v \) then \( e \Downarrow v \).

**Proof:** By induction on the proof that \( e \rightarrow e' \).

- Case: Rule 11. Then \( e = e_1 + e_2 \) and \( e' = e'_1 + e_2 \) and there is a subproof that \( e \rightarrow e' \). Also, if \( e' \) is an addition expression then \( e'_1 \Downarrow \overline{m_1} \) and \( e_2 \Downarrow \overline{m_2} \) and \( v = \overline{m_1 + m_2} \). (This is an inversion property for the \( \Downarrow \) relation.) By the inductive hypothesis, \( e_1 \Downarrow \overline{m_1} \). Therefore \( e \Downarrow \overline{m_1 + m_2} \) as required.
- Case: Rule 12: Analogous to previous case.
- Case: Rule 13: Then \( e = \overline{m_1} + \overline{m_2} \) and \( e' = v = \overline{m_1 + m_2} \). Obviously \( e \Downarrow v \).
- Case: Rules 14–16: Analogous to cases for addition.
- Case: Rule 17: Analogous to the case for rule 11.
- Case: Rule 18 and 19. Obvious by definition of \( \Downarrow \) that if \( e_2 \Downarrow v \) then if \( \texttt{true} \) then \( e_2 \) else \( e_3 \Downarrow v \), and similarly if \( e_3 \Downarrow v \) then if \( \texttt{false} \) then \( e_2 \) else \( e_3 \Downarrow v \).
- Case: Rule 20. Then \( e = (\texttt{let} \ x \ \texttt{be} \ e_1 \ \texttt{in} \ e_2) \) and \( e' = (\texttt{let} \ x \ \texttt{be} \ e'_1 \ \texttt{in} \ e_2) \) and \( e_1 \rightarrow e'_1 \) and \( e'_1 \Downarrow v_1 \) and \( e_2[x\rightarrow v_1] \Downarrow v \). By the inductive hypothesis, \( e_1 \Downarrow v_1 \). Thus \( \texttt{let} \ x \ \texttt{be} \ e_1 \ \texttt{in} \ e_2 \Downarrow v \).
- Case: Rule 21. Obvious by definition of \( \Downarrow \).
- Case: Rules 22 and 23. Analogous to cases for addition.
- Case: Rule 24. Obvious by definition of \( \Downarrow \).

3.

**Proposition 7**

If \( e \rightarrow^* v \) then \( e \Downarrow v \).

**Proof:** By induction on \( n \), the number of \( \rightarrow \) steps needed to reach \( v \) from \( e \).

- Case: \( n = 0 \). Then \( e = v \). Since \( e \) is a value, we have \( e \Downarrow v \) by Rule 35.
- Case: \( n > 0 \). Then \( e \rightarrow e' \rightarrow^* v \). By the inductive hypothesis, \( e' \Downarrow v \). By the previous part, then, \( e \Downarrow v \).