Do not turn this page until you have received the signal to start.
(In the meantime, please fill out the identification section above.)

This test consists of 5 questions on 5 pages (including this one).
When you receive the signal to start, please make sure that your copy of the test is complete.

Good Luck!
A program is a sequence of "top level" expressions and statements.

- **Expression Forms**
  - **Literal Value** \(\text{sn.sn sn/n} ; \text{number} \rightarrow \text{decimal or fractional form}\)
  - **image** \(--\text{characters}--\) ; text \#true \#false ; boolean
  - **list** literal-value etc ; list

- **Function name** by name from a definition or our language, or created anonymously ...
  - **Variable Reference** \(\text{name} \rightarrow \text{from a definition}\)
  - **Function Call** \(\text{function-name argument-expression etc}\)
  - **Parameter Reference** \(\text{name} \rightarrow \text{in the body of a function definition}\)
  - **Conditional** \(\text{if condition-expression consequent-expression else alternative-expression}\)

- **Statement Forms**
  - **Definition of Variable or Function**
    - \((\text{function-name value-expression})\)
    - \((\text{function-name parameter-name etc}) \rightarrow \text{"header" body-expression})\)
  - **Assertion / Test**
    - \((\text{same! expression expression (true! expression) (false! expression) etc})\)
  - **Inspect Evaluation**
    - \((\text{step expression})\)
    - \((\text{step (hide hide-option etc) expression})\)

- **Function-literal literal etc**
  - For a function from a definition, or an anonymous function:
    - If the number of arguments and parameter names differ, report an error.
    - If the name or the function call is a hide option and this is not the original call,
      then skip to the result value.
    - If the body is an if-expression, and if-introduction is a hide option, then skip to
      the consequent/alternative.
    - Otherwise: copy the function's body and substitute the arguments in place of
      the parameter names wherever those names occur in the body.
  - For the function map or combine: match its first pattern below, then:
    - If the expression doesn't match its pattern: report an error.
    - Otherwise: determine the literal values for \(a \ b \ c \ldots\), then substitute those
      into the rule's second pattern.

- **Function Design**
  - **Goal Example**
    - \((\text{same! (function-name argument etc) literal})\)
  - **Full Design**
    - \((\text{same! (function-name argument etc) fully-generalizable-expression})\)
  - **Partial Design**
    - \((\text{same! (function-name argument etc) partially-general-expression})\)

- **Function Examples**
  - **Equality Predicate**
    - \((\text{true! (same! (+ 1 1) 2)}) \ (\text{false! (same! 3 2)})\)
  - **Type Predicates**
    - \((\text{true! (image? \(\text{\(\ldots\)}\))}) \ (\text{true! (boolean? \(\text{false?}\))})\)
    - \((\text{true! (function? \(\text{flip}\))}) \ (\text{true! (text? \(\text{Hi!}\))})\)
    - \((\text{true! (number? \(-12.3\))}) \ (\text{true! (list? \(\text{\(\ldots\)}\))})\)
  - **Function Predicates**
    - \((\text{true! (unary? \(\text{flip}\))}) \ (\text{false! (binary? \(\text{flip}\))})\)
  - **Image Functions**
    - \((\text{same! (mirror \(\ldots\))})\)
    - \((\text{same! (scale-width 1.5)})\)
    - \((\text{same! (scale-height 1.5)})\)
    - \((\text{same! (circle 9)})\)
    - \((\text{same! (square 9)})\)
    - \((\text{same! (ove (circle 9) 7 15) })\)
    - \((\text{same! (rectangle 7 15)})\)
    - \((\text{same! (solid-triangle 9) \(\ldots\))})\)
    - \((\text{same! (solid-circle 9)})\)
    - \((\text{same! (solid-square 9)})\)
    - \((\text{same! (solid-oval 7 15)})\)
    - \((\text{same! (solid-rectangle 7 15) \(\ldots\))})\)
    - \((\text{same! (width (oval 7 15) 7)})\)
    - \((\text{same! (height (oval 7 15) 15)})\)
  - **Numeric Functions**
    - \((\text{same! (+ 2 10 3) 15})\)
    - \((\text{same! (- 12 12) -12})\)
    - \((\text{same! (/ 12 3) 4})\)
    - \((\text{same! (* 2 10 3) 60})\)
    - \((\text{same! (number->text -12) \"-12\")})\)
  - **Text Functions**
    - \((\text{same! (text-length \"one\") 3})\)
    - \((\text{same! (text-join \"Hi\" \"human\" \\"Hi human\")})\)
    - \((\text{same! (text->image \"Hi\") \(\ldots\) \(\ldots\))})\)
    - \((\text{same! (text->list \"Hi\") \(\\ldots\) \(\\ldots\) \(\\ldots\))})\)
  - **List Functions**
    - \((\text{same! (list (star 10) (+ 2 3) (text? 4) (list \(\ldots\) \#false))})\)
    - \((\text{same! (length (list \(\ldots\) \#false)) 3})\)
    - \((\text{same! (rest (list \(\ldots\) \#false)) \(\ldots\))})\)
    - \((\text{same! (reverse (list \(\ldots\) \#false)) \(\ldots\))})\)
    - \((\text{same! (select number? (list \(\ldots\) \#false)) (list 5)})\)
    - \((\text{same! (range 8) \(\ldots\)) \(\ldots\))\)
    - \((\text{same! (range 3 8) \(\ldots\)) \(\ldots\))\)
    - \((\text{same! (range 3 8 2) \(\ldots\)) \(\ldots\))\)
Question 1 [ 5 + 2 = 7 marks ]

Show the following steps, with standard underlining...

(step (map ((fun t) (same? (text-length t) 3)))
  (list "ant" "bear" "cod"))

Question 2 [ 5 marks ]

(define (T f a-list)
  (map ((fun e) (list e (f e)))
       a-list))

Show the following steps, with standard underlining...

(step (T triangle (list 20 30 10)))
★★ Question 3  [ 4 + 4 = 8 marks ]

; Here is an assertion for a new function point ...

\[ (\text{same! } (\text{point } "\text{cats}" \ (\text{list } \square \square \square)) \ ) \]

; ✓ Insert a Full Design expression into the following assertion for point ...

\[ (\text{same! } (\text{point } "\text{happy}" \ (\text{list } \square \square \square)) \ ) \]

\[ (\text{beside-top } (\text{text->image } "\text{happy}: ")) \]

\[ (\text{above-left } \square \square \square) \]

\[ \text{happy: } \]

\[ \square \]

\[ \square \]

; ✓ Define point ...

★★ Question 4  [ 2 + 3 + 3 = 8 marks ]

; ✓ Show the values of the following expressions ...

\[ (\text{length } (\text{list } "\text{shapes}" \square \square \square) \ ) \]

\[ (\text{list } "\text{solid}" \ (\text{list } "\text{pointy"} \square \ ▲ \ ●) \ △) \]

\[ (\text{reverse } (\text{list } "\text{shapes}" \square \square \square) \ ) \]

\[ (\text{list } "\text{solid}" \ (\text{list } "\text{pointy"} \square \ ▲ \ ●) \ △) \]

\[ (\text{select image? } (\text{list } "\text{shapes}" \square \square \square) \ ) \]

\[ (\text{list } "\text{solid}" \ (\text{list } "\text{pointy"} \square \ ▲ \ ●) \ △) \]
★ Question 5 \[ 3 + 6 + 4 + 2 = 15 \text{ marks} \]

; Consider the following function \texttt{out}, which uses the function \texttt{point} from Question 3 ...

\begin{verbatim}
define (out v)
  (if (list? v) (point (first v) (map out (rest v)))
   else v))
\end{verbatim}

; ✓ Show the following three sets of steps, with standard underlining.
; But you may include any skipped steps that help you.

\begin{verbatim}
(step (out ▲))
\end{verbatim}

\begin{verbatim}
(step (hide if-conditions point (out "pointy") (out ■) (out ▲)) (out (list "pointy" ■ ▲)))
\end{verbatim}

; Recall: (hide if-conditions \cdots) lets you skip showing any step of the form (if #true \cdots else \cdots) or (if #false \cdots else \cdots).

\begin{verbatim}
(step (hide if-introduction point out) (out (list "solid" (list "pointy" ■ ▲) ●)))
\end{verbatim}

; Recall: (hide if-introduction \cdots) lets you skip showing any step of the form (if \cdots else \cdots).

; Recall: (hide \cdots out) lets you skip showing the steps to produce the result of all recursive calls (out \cdots).

\begin{verbatim}
(step (hide if-condition point out) (out (list "solid" (list "pointy" ■ ▲) ●)))
\end{verbatim}

; ✓ Show the value of the following expression ...

\begin{verbatim}
(out (list "shapes" ◯ □ (list "solid" (list "pointy" ■ ▲) ●) △))
\end{verbatim}