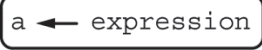

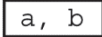




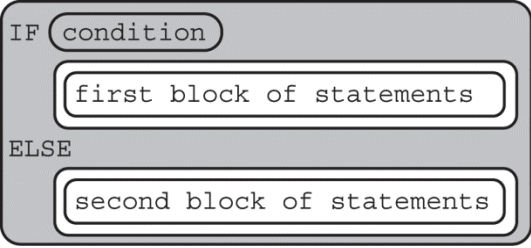




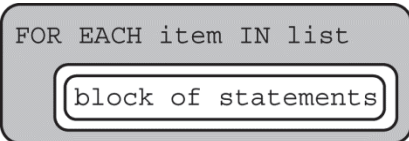
## REFERENCE MATERIALS

Instruction	Explanation
<b>Assignment, Display, and Input</b>	
<p>Text: a ← expression</p> <p>Block: </p>	Evaluates expression and assigns the result to the variable a.
<p>Text: DISPLAY (expression)</p> <p>Block: </p>	Displays the value of expression, followed by a space.
<p>Text: INPUT ()</p> <p>Block: INPUT</p>	Accepts a value from the user and returns it.
<b>Arithmetic Operators and Numeric Procedures</b>	
<p>Text and Block: a + b a - b a * b a / b</p>	<p>The arithmetic operators +, -, *, and / are used to perform arithmetic on a and b.</p> <p>For example, 3 / 2 evaluates to 1.5.</p>
<p>Text and Block: a MOD b</p>	<p>Evaluates to the remainder when a is divided by b. Assume that a and b are positive integers.</p> <p>For example, 17 MOD 5 evaluates to 2.</p>
<p>Text: RANDOM (a, b)</p> <p>Block: RANDOM </p>	<p>Evaluates to a random integer from a to b, including a and b.</p> <p>For example, RANDOM (1, 3) could evaluate to 1, 2, or 3.</p>
<b>Relational and Boolean Operators</b>	
<p>Text and Block: a = b a ≠ b a &gt; b a &lt; b a ≥ b a ≤ b</p>	<p>The relational operators =, ≠, &gt;, &lt;, ≥, and ≤ are used to test the relationship between two variables, expressions, or values.</p> <p>For example, a = b evaluates to true if a and b are equal; otherwise it evaluates to false.</p>
<p>Text: NOT condition</p> <p>Block: NOT </p>	Evaluates to true if condition is false; otherwise evaluates to false.
<p>Text: condition1 AND condition2</p> <p>Block: </p>	Evaluates to true if both condition1 and condition2 are true; otherwise evaluates to false.

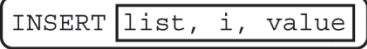
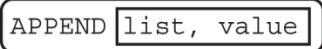


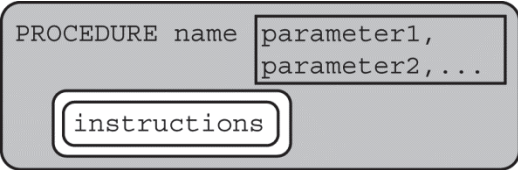
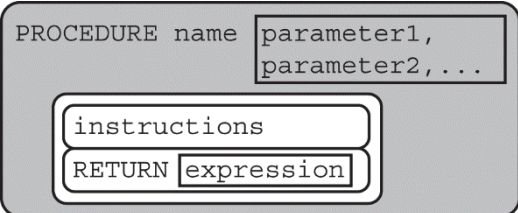
## REFERENCE MATERIALS

Instruction	Explanation
<b>Relational and Boolean Operators (continued)</b>	
<p><b>Text:</b> condition1 OR condition2</p> <p><b>Block:</b>  </p>	<p>Evaluates to true if condition1 is true or if condition2 is true or if both condition1 and condition2 are true; otherwise evaluates to false.</p>
<b>Selection</b>	
<p><b>Text:</b> IF (condition) {   &lt;block of statements&gt; }</p> <p><b>Block:</b>  </p>	<p>The code in block of statements is executed if the Boolean expression condition evaluates to true; no action is taken if condition evaluates to false.</p>
<p><b>Text:</b> IF (condition) {   &lt;first block of statements&gt; } ELSE {   &lt;second block of statements&gt; }</p> <p><b>Block:</b>  </p>	<p>The code in first block of statements is executed if the Boolean expression condition evaluates to true; otherwise the code in second block of statements is executed.</p>
<b>Iteration</b>	
<p><b>Text:</b> REPEAT n TIMES {   &lt;block of statements&gt; }</p> <p><b>Block:</b>  </p>	<p>The code in block of statements is executed n times.</p>

## REFERENCE MATERIALS

Instruction	Explanation
<b>Iteration (continued)</b>	
<p>Text:  REPEAT UNTIL (condition)  {      &lt;block of statements&gt;  }</p> <p>Block:</p> 	<p>The code in block of statements is repeated until the Boolean expression condition evaluates to true.</p>
<b>List Operations</b>	
<p>For all list operations, if a list index is less than 1 or greater than the length of the list, an error message is produced and the program terminates.</p>	
<p>Text:  list[i]</p> <p>Block:  list <span style="border: 1px solid black; padding: 0 2px;">i</span></p>	<p>Refers to the element of list at index i. The first element of list is at index 1.</p>
<p>Text:  list[i] ← list[j]</p> <p>Block:  <span style="border: 1px solid black; padding: 2px;">list <span style="border: 1px solid black; padding: 0 2px;">i</span> ← list <span style="border: 1px solid black; padding: 0 2px;">j</span></span></p>	<p>Assigns the value of list[j] to list[i].</p>
<p>Text:  list ← [value1, value2, value3]</p> <p>Block:  <span style="border: 1px solid black; padding: 2px;">list ← <span style="border: 1px solid black; padding: 2px;">value1, value2, value3</span></span></p>	<p>Assigns value1, value2, and value3 to list[1], list[2], and list[3], respectively.</p>
<p>Text:  FOR EACH item IN list  {      &lt;block of statements&gt;  }</p> <p>Block:</p> 	<p>The variable item is assigned the value of each element of list sequentially, in order from the first element to the last element. The code in block of statements is executed once for each assignment of item.</p>

## REFERENCE MATERIALS

Instruction	Explanation
<b>List Operations (continued)</b>	
<p>Text: INSERT (list, i, value)</p> <p>Block:  </p>	<p>Any values in list at indices greater than or equal to i are shifted to the right. The length of list is increased by 1, and value is placed at index i in list.</p>
<p>Text: APPEND (list, value)</p> <p>Block:  </p>	<p>The length of list is increased by 1, and value is placed at the end of list.</p>
<p>Text: REMOVE (list, i)</p> <p>Block:  </p>	<p>Removes the item at index i in list and shifts to the left any values at indices greater than i. The length of list is decreased by 1.</p>
<p>Text: LENGTH (list)</p> <p>Block:  </p>	<p>Evaluates to the number of elements in list.</p>
<b>Procedures</b>	
<p>Text: PROCEDURE name (parameter1,                                   parameter2, ...)</p> <pre>{     &lt;instructions&gt; }</pre> <p>Block:  </p>	<p>A procedure, name, takes zero or more parameters. The procedure contains programming instructions.</p>
<p>Text: PROCEDURE name (parameter1,                                   parameter2, ...)</p> <pre>{     &lt;instructions&gt;     RETURN (expression) }</pre> <p>Block:  </p>	<p>A procedure, name, takes zero or more parameters. The procedure contains programming instructions and returns the value of expression. The RETURN statement may appear at any point inside the procedure and causes an immediate return from the procedure back to the calling program.</p>

## REFERENCE MATERIALS

Instruction	Explanation
<b>Robot</b>	
If the robot attempts to move to a square that is not open or is beyond the edge of the grid, the robot will stay in its current location and the program will terminate.	
Text: MOVE_FORWARD ()  Block: <div style="border: 1px solid black; border-radius: 10px; padding: 2px; display: inline-block;">MOVE_FORWARD</div>	The robot moves one square forward in the direction it is facing.
Text: ROTATE_LEFT ()  Block: <div style="border: 1px solid black; border-radius: 10px; padding: 2px; display: inline-block;">ROTATE_LEFT</div>	The robot rotates in place 90 degrees counterclockwise (i.e., makes an in-place left turn).
Text: ROTATE_RIGHT ()  Block: <div style="border: 1px solid black; border-radius: 10px; padding: 2px; display: inline-block;">ROTATE_RIGHT</div>	The robot rotates in place 90 degrees clockwise (i.e., makes an in-place right turn).
Text: CAN_MOVE (direction)  Block: CAN_MOVE <span style="border: 1px solid black; padding: 0 5px;">direction</span>	Evaluates to <code>true</code> if there is an open square one square in the direction relative to where the robot is facing; otherwise evaluates to <code>false</code> . The value of direction can be <code>left</code> , <code>right</code> , <code>forward</code> , or <code>backward</code> .