exceptions. Its definition is straightforward: We needed the environment recovery operation twice in our semantics above, the first time for blocks and the second time for exceptions. 

The simplest way to give the semantics of the while loop is by unrolling. Note, however, that its unrolling is only allowed in the conventional meaning of a computation task of the form \( \text{while } \{ \text{stmts} \} \). The rules below cover the only two possibilities in which the conditional is allowed to proceed (otherwise the body of the while loop is simply evaluated in its desugaring form). This is due to the fact that we did not define control statements in SIMPLE.

We assume that \( \text{stmts} \) is a list of statements with one \( \text{stmt} \) as its head. The \( \text{stmt} \) can be any of the following forms: assignment, expression, conditional, loop, function call, or other control statements.

### Assignment

The assignment \( \text{stmt} = \text{expr} \) is evaluated as follows: We first evaluate the right-hand side \( \text{expr} \) and then update the environment with the value \( \text{expr} \) and the corresponding name of the left-hand side variable. The \( \text{stmt} \) is evaluated in the desugaring code. Note that both the loop condition and the nested multi-dimensional declaration of the array are evaluated.

### Expression Statements

Expression statements are only used for their side effects, so their result value is simply discarded. Common examples of expression statements are: function calls, assignment statements, \( \text{input} \) and \( \text{output} \) statements, and \( \text{spawn} \) or \( \text{block} \) statements.

### Conditional Statements

A conditional statement of the form \( \text{cond} \text{stmt} \) is evaluated as follows: We first evaluate the condition \( \text{cond} \) and check if it is true. If it is true, we evaluate the \( \text{stmt} \) and return its result value. If the condition is false, we return the result value of the \( \text{stmt} \) associated with the \( \text{false} \) branch.

### Loop Statements

Loop statements like \( \text{while} \{ \text{stmts} \} \) are evaluated in the conventional meaning of a computation task of the form \( \text{while} \). The loop is executed as long as the condition is true. The loop continues to execute until the condition becomes false. The body of the loop is evaluated in the desugaring code.

### Function Statements

Function statements like \( \text{function} \{ \text{stmts} \} \) are evaluated in the conventional meaning of a computation task of the form \( \text{function} \). The function is defined at the top level and can be called from any point in the program. The function body is evaluated in the desugaring code.

### Environment Recovery

Environment recovery is a crucial aspect of the semantics of SIMPLE. The environment recovery operation is used to update the environment after a statement has been evaluated. The operation consists of three steps: 1) updating the \( \text{env} \) variable of the current thread, 2) updating the \( \text{genv} \) variable of the current thread, and 3) updating the \( \text{fstack} \) variable of the current thread.

### Configuration

The configuration of SIMPLE consists of a top level cell, \( \text{config} \), and a list of \( \text{stmts} \). The configuration is the only part of the program that is visible to the user. The \( \text{config} \) variable is used to store the current configuration of the program. The \( \text{stmts} \) variable is used to store the list of statements that are currently being executed.

### Variable Lookup

Variable lookup is a crucial aspect of the semantics of SIMPLE. The variable lookup operation is used to retrieve the value of a variable from the environment. The operation consists of three steps: 1) updating the \( \text{env} \) variable of the current thread, 2) updating the \( \text{genv} \) variable of the current thread, and 3) updating the \( \text{fstack} \) variable of the current thread.