3AC Language Specification (for this interpreter)

The interpreter expects 3AC instructions in a line-by-line format. Labels are followed by a colon. Operands can be variable names, literal numbers, or boolean values.

Basic Operations:

- LABEL <name>:: Defines a jump target.
- ASSIGN <target>, <source>: target = source (variable or literal).
- CONST_ASSIGN <target>, <value>: target = value (literal value).
- ADD <target>, <op1>, <op2>: target = op1 + op2.
- SUB <target>, <op1>, <op2>: target = op1 - op2.
- MUL <target>, <op1>, <op2>: target = op1 * op2.
- DIV <target>, <op1>, <op2>: target = op1 / op2.
- MOD <target>, <op1>, <op2>: target = op1 % op2.
- EQ <target>, <op1>, <op2>: target = (op1 == op2).
- NE <target>, <op1>, <op2>: target = (op1 != op2).
- LT <target>, <op1>, <op2>: target = (op1 < op2).
- LE <target>, <op1>, <op2>: target = (op1 <= op2).
- GT <target>, <op1>, <op2>: target = (op1 > op2).
- GE <target>, <op1>, <op2>: target = (op1 >= op2).
- JUMP <label>: Unconditional jump to label.
- JUMPT <label>, <condition_var>: Jump to label if condition_var is True.
- JUMPF <label>, <condition_var>: Jump to label if condition_var is False.
- PRINT <value>: Prints the value to the console.
- HALT: Stops program execution.

Function Call Operations:

- PARAM <value>: Pushes value as an argument for the next function call.
- CALL <func_name>, <num_params>, <return_var>:
  Calls func_name with num_params arguments (from PARAMs). The return value (if any) is stored in return_var.
- RETURN <value>: Returns value from the current function.

Pointer and Heap Operations (for ADTs and dynamic data):

The heap is a collection of "memory cells" accessed by integer addresses. ADTs are blocks of these cells.

- ALLOC_HEAP <target_ptr_var>, <size>: Allocates size contiguous memory cells on the heap and stores the starting address (an integer) in target_ptr_var.
- FREE_HEAP <ptr_var>: (Simplified) Marks memory pointed to by ptr_var as available. (Actual memory reclamation is complex and omitted for simplicity).
- ADDR_OF <target_ptr_var>, <source_var>:
    - If source_var is a global or local variable, its current value is copied to a *new* heap location, and the address of that heap location is stored in target_ptr_var.
    - If source_var itself holds a heap address (i.e., it's already a pointer), target_ptr_var will get that same address.
- DEREF_LOAD <target_var>, <ptr_var>: target_var = *ptr_var (loads the value from the heap address stored in ptr_var).
- DEREF_STORE <ptr_var>, <value_var>: *ptr_var = value_var (stores value_var at the heap address stored in ptr_var).
- INDEX_LOAD <target_var>, <base_ptr_var>, <index_var>: target_var = base_ptr_var[index_var] (loads value from heap[base_ptr_var + index_var]).
- INDEX_STORE <base_ptr_var>, <index_var>, <value_var>: base_ptr_var[index_var] = value_var (stores value_var at heap[base_ptr_var + index_var]).