Phase 3 Documentation

All changes were made with reference to phase 3 documentation and project description. Documentation for the tests is available for each test in the /testSuite/phase3 directory.

Change made	**Location**	**Explanation/Justification**
Phase 2 fixes	'LikeClause' and 'ValueOrLike' rules in parser.ssl	🍹 Replaced the logic for the 'like' keyword to disallow an array of files. 🍹 Added an expected ';' 'Block' rule after the call to 'RepeatStmt' to end the call.
Input tokens	Input tokens in semantic.ssl	🍹 Copied the contents of the parser.ssl Output tokens into the Input tokens of semantic.ssl.
Output tokens	Output tokens in semantic.ssl	🍹 Added the T-code outputs (tConcatenate, tRepeatString, tSubstring, tLength, tStringEQ, tInitialValue, iInitEnd, tCaseElse, tCaseElseEnd) to Output tokens' in semantic.ssl.
Programs	semantic.ssl	🍹 By updating the the Input tokens in semantic.ssl and removing all references to old PT Pascal tokens (sType), the necessary changes for Like programs were complete
Blocks and Statements	'Block', 'Statement', 'BeginStmt' in semantic.ssl	🍹 Added 'sBegin' expected input token at the beginning of 'Block'. 🍹 Removed the case for 'sType' from the choices in 'Block'. 🍹 Added cases from 'Statement' into 'Block'. 🍹 Removed the expected 'sBegin' input token at the end of 'Block' and the call to 'BeginStmt'. 🍹 Added an expected 'sEnd' input token at the end of 'Block'. 🍹 Removed the choice block from 'Statement' rule. 🍹 Changed 'Statement' rule to push a new scope to the symbol table, call the 'Block' rule, and pop the scope from the symbol table.
Variables and Types	'TypeDefinitions', 'VariableDeclarations;, 'SimpleType', 'IndexType', 'ProcedureParameterType', 'TypeBody' in semantic.ssl	🍹 'TypeDefinitions' rule removed since Like has no type declarations 🍹 'VariableDeclarations' no longer loops so it only allows one variable per **var** declaration 🥥 Also now checks for sPublic token (more details in Packages section) 🥥 Handles the case of an initial value (more details below) 🍹 'SimpleType' changed to begin with sLike and have alternatives for sInteger, sIdentifier, and sStringLiteral 🥥 Removed choice for sRange as Like does not use ranges in its variable declarations 🥥 Since a like clause accepts an optionally negated integer, added a check for the sNegate token, which is simply ignored 🥥 Chose tpInteger to be the type pushed to the Type Stack if the symbol kind is invalid

		🍹 ‘IndexType’ changed to push the constant 1 as the lower bound and use the ‘ConstantValue’ rule to push the upper bound 🥥 Removed the call to ‘SimpleType’ as array bounds in Like do not determine type 🍹 ‘ProcedureParameterType’ changed to simply call ‘SimpleType’ before calling ‘AllocateVar’ as formal parameters in Like are specified using the like clause 🍹 ‘TypeBody’ rule modified to check for sFile token and throw an error in the case of the attempted declaration of an array of files, which is not supported in Like
Initial Values	‘VariableDeclarations’ in semantic.ssl	🍹 ‘VariableDeclarations’ rule changed to handle an initial value declaration 🥥 ‘Expression’ rule used to accept the value of an expression and push its attributes to the Type and Symbol Stacks 🥥 These attributes are then copied to the new variable using ‘EnterVariableAttributes’ 🥥 Emit a reverse assignment of the value to the variable 🥥 Emit .tInitialValue and .tInitEnd tokens before and after the call to ‘Expression’ as these are necessary T-code instructions for the Like abstract machine
Packages	semantic.ssl	🍹 Added oSymbolTblMergePublicScope to Symbol Table mechanisms 🍹 Added oSymbolStkSetPublicFlag to Symbol Stak mechanisms 🍹 Added new type ‘syPackage’ to SymbolKind 🍹 Added new IsPublic rule that checks for sPublic token and calls oSymbolStkSetPublicFlag 🍹 Added new PackageDefinition rule 🥥 Called from Block rule and is similar to the ProcedureDefinition rule 🥥 Pushes the package name and type (‘syPackage’) to the symbol stack 🥥 Push the package scope to the symbol table 🥥 Calls Block rule to continue handling enclosed statements before update the type table 🥥 Call oSymbolTblMergePublicScope to transfer variables with public scope to the enclosing scope (more details below in following section) 🍹 Check for \`sPackage\` in Block rule and call PackageDefinition rule 🍹 Use IsPublic rule to check for sPublic tokens in ConstantDefinitons, VariableDeclarations, and ProcedureDefinition rules
Packages	semantic.pt	🍹 Added a ‘symbolTblPublicFlag’ to the symbol table (boolean array) 🍹 Added a ‘symbolStkPublicFlag’ to the symbol stack (boolean array) 🍹 ‘symbolStkPublicFlag’ is initialized to false in SymbolStkPush procedure 🍹 ‘symbolStkPublicFlag’ is set from the symbol table in SymbolStkPushIdentifier procedure 🍹 ‘symbolTblPublicFlag’ is set from the symbol stack when adding an entry to the symbol table with oSymbolTblEnter mechanism

		🍹 ‘symbolTblPublicFlag’ is set from the symbol stack when updating an entry in the symbol table with oSymbolTblUpdate mechanism 🍹 Added a new mechanism oSymbolTblMergePublicScope to merge variables with public keyword to the enclosing scope by checking the ‘symbolTblPublicFlag’ - entries in the symbol table with the flag set to false are unlinked (popped) from the type table 🍹 Added a new mechanism oSymbolStkSetPublicFlag that sets ‘symbolStkPublicFlag’ to true
Statements (short form assignment, repeat, while, ifs, choose)	‘CaseStmt’ in semantic.ssl	🍹 In the ‘CaseStmt’ rule, added a choice for an optional ‘sCaseElse’ token that emits ‘tCaseElse’, calls the ‘Statement’ rule, and emits ‘tCaseElseEnd’. For any other case, do not perform any actions.
Reuse PTs char type T-codes for Strings in Like	Semantic.ssl & Semantic.pt	🍹 Added definition for stringSize = 256 in type Integer to be used when allocating storage to store Like strings 🍹 Removed ‘tpString’ from TypeKind as Strings in Like are to be handled with ‘tpChar’ 🍹 Removed all case alternatives that use ‘tpString’ in semantic.ssl since strings in Like will be handled with ‘tpChar’ 🥥 ‘CompareAndSwapTypes’, ‘ConstantOperand’, ‘AssignProcedure’, and ‘WriteText’ rules in semantic.ssl
String Constants and Variables	‘Operand’, ‘ConstantOperand’, and StringLiteral“ in semantic.ssl	🍹 In ‘Operand’, changed T-code emitted sStringLiteral to be tLiteralAddress followed by a tFetchChar 🍹 Removed oTypeStkChooseKind case inside sStringLiteral since there's no difference between chars and strings in Like 🍹 In ‘ConstantOperand’, changed ‘tpChar’ to emit ‘.tLiteralAddress’ instead of ‘.tLiteralChar’ 🍹 In ‘StringLiteral’, removed the handling of length zero and length one characters since they are not needed when using Like strings 🍹 Change PT’s character array construction with a push of ‘tpChar’ (Like String) onto the type stack and link it to stdChar
String Traps	‘TrapKind’, ‘WriteText and ‘ReadText’ in semantic.ssl	🍹 Input and output of Like strings will use the new Like trap codes trWriteString and trReadString which have trap values 109 and 108 respectively 🍹 In type TrapKind, changed ‘trWriteString’ to have the value 109 and add’ trReadString’ = 108 🍹 In WriteText rule, changed the trap emitted from trWriteChar to trWriteString in the tpChar case alternative 🍹 In ReadText rule, changed the trap emitted from trReadChar to trReadString in the tpChar

		case alternative
String Allocation	‘oValuePushChar’, ‘oAllocateVariable’ ‘oEmitString’ semantic operations in semantic.pt	🍹 Changed ‘oValuePushChar’ to push the code address of the string literal (codeAreaEnd) instead of 1 since characters can now be more than 1 character 🍹 Changed how ‘oAllocateVariable’ handles ‘tpChar’ to now allocate 256 bytes instead of 1 🍹 Changed how ‘oAllocateVariable’ handles ‘tpArray’ to now allocate 256*stringSize when the array is of kind ‘tpChar’ 🍹 Changed ‘oEmitString’ to emit an ASCII null character (0) since String Literals are of varying lengths
String equality operations (equality, inequality, ordering)	‘BinaryOperator’ and ‘CompareEqualityOperandTypes’ in semantic.ssl	🍹 In ‘CompareEqualityOperandTypes’, added the T-code to emit ‘tStringEQ’ if the top of the type stack contains an array of chars (a string). For all other cases, emit ‘tEQ’. 🍹 In ‘BinaryOperator’, removed the emission of ‘tEQ’ and ‘tNE’ for the case of ‘sEq’ and ‘sNE’ from the input token stream. For the sNE case, after the call to ‘CompareEqualityOperandTypes’, emit .tNot. 🍹 Note: this removes the need for the T-code output tNE.
Other string operations (length, concatenate, substring, repeat string)	‘UnaryOperator’ and ‘BinaryOperator’ in semantic.ssl	🍹 Added a case for ‘sLength’ input token in ‘UnaryOperator’. Instead of calling ‘CompareAndSwapTypes’, ensure that the input type is a char array, and the result type is an integer. 🍹 Added a case for ‘sConcatenate’ input token in ‘BinaryOperator’. Push ‘tpChar’ to the type stack to ensure the result type is an array of chars, and call ‘CompareOperandAndResultTypes’ to ensure the input types are char arrays as well. 🍹 Added a case for ‘sRepeatString’ input token in ‘BinaryOperator’. Push ‘tpChar’ to the type stack to ensure the result type in an array of chars. Ensure the top of the type stack (the second term in the expression) is an integer representing the amount of times to repeat. Then call ‘CompareAndSwapTypes’ to ensure the first term in the repeat string expression is an array of chars. 🍹 Added a case for ‘sSubstring’ input token in ‘BinaryOperator’. Push ‘tpChar’ to the type stack to ensure the result type is an array of chars. Check that the top two types on the stack (the second and third term in the expression) are integer constants or variables. Call ‘CompareAndSwapTypes’ to ensure the first term in the substring expression is a string literal constant or variable.