Project Checklist

1.1 input interface: accepts user-provided file name as a parameter, as opposed to a hard-coded file name 1 Done via console args

* 1. output interface: clarity/usefulness of standard output, clarity/usefulness of alternate output to different files 2 Done via IOHelper, printing to console and specified file
  2. all errors are reported in a single stream in synchronized order, even if errors are found in different phases 2 Done via list of errors per section.

2.1.1 integers and floating point numbers (valid/invalid numbers according to assignment 1 handout) 1 Done in Lexical Analyzer, see LexicalAnalyzerTests

2.1.2 comments: inline comments, block comments, unending block comments, nested block comments 1 Done but Nested block comments don’t work

2.2.1 lexical error detection: detecting all lexical errors in a program 1 Done see outlexerrors

2.2.2 lexical error reporting: accurate reporting of errors in a .outlexerrors file, including line number and useful description of the error 1 Done see outlexerrors

2.3.1 output of token stream in a .outlextokens output file 1 Done see outlextokens

3.1.1 variable declarations: int, float, class types, array, array of class types 1 Done see derivationtree/ast

3.1.2 main function 1 Done see derivationtree/ast

3.1.3 free functions 2 Done see derivationtree/ast

3.1.4 member function definitions 2 Done see derivationtree/ast

3.1.5 class declarations: data member declarations, method declarations, inheritance list 2 Done see derivationtree/ast

3.1.6 complex expressions (all arithmetic, relational and logic operators in one expression) 2 Need to Test mostly done see derivationtree/ast

3.1.7 conditional statement, including nested if without brackets 2 Done see derivationtree/ast

3.1.8 loop statement, including nested for without brackets 2 Done see derivationtree/ast

3.1.9 read(var) / write(expression) / return(expression) statements 1 Done see derivationtree/ast

3.1.10 access to class members, including multiply nested and including array members 2 Done see derivationtree/ast

3.1.11 access to arrays: uni- and multi-dimensional, using expressions as index 2 Done see derivationtree/ast

3.2.1 syntax detection: detecting all syntax errors in a program 1 Done see outsyntaxerrors

3.2.2 syntax error reporting: accurate reporting of errors in a .outsyntaxerrors file including line number and useful description of the error 1 Done see outsyntaxerrors

3.2.3 syntax error recovery: implementation of an effective syntax error recovery mechanism 2 Done see syntactic analysis freshen up on terminology and method

3.3.1 generation of an AST 3 Done but not perfect due to specifications, due to not using visitor pattern

3.3.2 output a derivation of the compiled program in a .outderivation output file 2 Done

3.3.3 output the AST of the compiled program in a .outast output file 2 Done

4.1.1 AST tree traversal that triggers semantic actions 4 Done but not using visitor pattern

4.1.2 global scope symbol table 1Done

4.1.3 entry in the global table for each class declared. Local tables for classes 1 Done

4.1.4 entry in the appropriate table for each function definition (free functions and member functions). Local tables for each function 1 Done for free, partially for member functions

4.1.5 entry in the appropriate table for each variable defined in a scope, i.e. class data members and function’s local variables 1 Done

4.2.1 type checking on large expressions, assignment and return statements 3 Done see semanticTest1

4.2.2 operators not allowed on objects 1 Done see semanticTest2

4.2.3 checking of type and number of parameters upon a function call (free functions and member functions) 2 Done see semanticTest3

4.2.4 use of an array variable made using the same number of dimensions as declared in the variable declaration 1 Done see semanticTest4

4.2.5 expressions used as an index must be of integer type 1 Done see semanticTest5

4.2.6 circular class dependencies (through data members or inheritance) are detected and not allowed 2 Done see semanticTest6

4.2.7 the “.” operator used only on variables of a class type (maybe same as undeclared member) 1 Done see semanticTest7

4.2.8 forward references for classes/free functions 1 Done see semanticTest7

4.2.9 undeclared function: definition or call to a function that is not declared (free function or member function) 1 Done see semanticTest7

4.2.10 undefined function: declaring a member function that does not have a corresponding function definition 1 Not done

4.2.11 undefined class 1 Done see semanticTest7

4.2.12 missing return statement 1 Done see semanticTest8

4.2.13 mismatch between member function declaration and definition 1 Not done

4.2.14 warning for shadowed data members upon inheritance 1 Done see semanticTest9

4.2.15 member function defined as part of non-existing class 1 Not Done

4.2.16 undeclared variable: use of a local variable name for which there is no declaration 1 Done see semanticTest7

4.2.17 undeclared data member: reference to a data member that is not declared (including in superclasses or deeply nested) 2 Done see semanticTest9

4.2.18 multiply declared variable: an identifier cannot be declared twice in the same scope 1 Done see semanticTest9

4.3.1 semantic error detection: detecting all semantic errors in a program 1 Done

4.3.2 semantic error reporting: accurate reporting of errors, including line number and useful description of the error 1 Done however does not include number line anymore, due to bad decisions in tree structure

4.4.1 output of the symbol table structure of the compiled program in a .outsymboltables output file 2 Done see file

5.1.1 allocate memory for basic types (integer, float) 1 Done see allocationTest

5.1.2 allocate memory for arrays of basic types 1 Done see allocationTest

5.1.3 allocate memory for objects 1 Done see allocationTest

5.1.4 allocate memory for objects with inheritance 1 Done see allocationTest

5.1.5 allocate memory for objects having object members 1 Done see classTest2

5.1.6 allocate memory for arrays of objects 1 Done see allocationTest

5.1.7 allocate memory for temporary results 1 Done see expressionTest

5.2.1 branch to a function’s code block, execute the code block, branch back to the calling function upon return 2 Done see functionTest

5.2.2 branch back to a function that has been branched upon 1 Done for write and read see functionTest

5.2.3 pass parameters as local values to the function’s code block 2 Done see functionTest

5.2.4 upon function resolution, pass the return value back to the calling function 1 Done see functionTest

5.2.5 function call stack mechanism 2 Not done

5.2.6 call to member functions 2 Not done

5.3.1 assignment statement: assignment of the resulting value of an expression to a variable, independently of what is the complexity of the expression 1 Done see any tests

5.3.2 conditional statement: implementation of branching mechanism, including for imbricated conditional statements 2 Done see ifElseTest and innerIfTest

5.3.3 loop statement: correct implementation of branching mechanism, including for imbricated loop statements 2 Done see loopTest and innerLoopTest

5.3.4 input/output statements: read()/write() 2 Done see any test for write and readTest

5.4.1 arrays of basic types (integer and float), access to an array’s elements, single or multidimensional 1

Done for single and not multi, see arrayTest

5.4.2 arrays of objects, access to an array’s object elements, single or multidimensional 1 Done for single and not multi, arrayTest

5.4.3 objects, access to members of basic types 1 Done see classTests

5.4.4 objects, access to members of array types, as well as the elements of the array 1 Done see classTests

5.4.5 objects, access to members of object types, as well as the elements of the object 1 Done see classTests

5.4.6 objects, access to the members of a superclass 1 Not done

5.5.1 computing the value of an entire complex expression involving all of: arithmetic, relational and logic operators in one expression 2 Mostly done see expressionTest Test and or not

5.5.2 expression involving an array factor whose indexes are themselves expressions 1 Done see arrayTest

5.5.3 expression involving an object factor referring to object members (dot operator) 1 Done see classTests

5.6.1 output of the generated code of the compiled program in a .moon output file 1 Done see .moons