A	A	1.1	Allocate memory for basic types (integer, float).
A	A	1.2	Allocate memory for arrays of basic types.
A	A	1.3	Allocate memory for arrays of objects.
A	A	2.1	Branch to a function's code block, execute the code block, branch back to the calling function.
A	A	2.2	Pass parameters as local values to the function's code block.
A	A	2.3	Upon execution of a return statement, pass the return value back to the calling function.
A	A	2.4	Call to member functions that can use their object's data members.
A	A	3.1	Assignment statement: assignment of the resulting value of an expression to a variable, independently of what is the expression to the right of the assignment operator.
A	A	3.2	Conditional statement: implementation of a branching mechanism.
A	A	3.3	Loop statement: implementation of a branching mechanism.
A	A	3.4	Input/Output statement: Move machine keyboard input/console output
A	A	5.1	For array of basic types (integer and float), access to an array's elements.
A	A	5.2	For array of objects, access to an array's element's data members.
A	A	5.3	For objects, access to members of basic types.
A	A	5.4	For objects, access to members of array or object types.
A	A	5.1	Computing the value of an entire complex expression.
A	A	5.2	Expression involving an array factor whose indexes are themselves expressions.
A	A	5.3	Expression involving an object factor referring to object members.

Notes	%mark	grade	unweighted	letter		
	100%		50			
	100%	2	2		1	Document Section 1 - Checklist of code generation items implemented, along with mapping to test files.
	100%	2	2	A	1.1	Using the (1–5) itemized list provided above (see "Implementation"), provide a checklist that identifies what code generation items either implemented or not implemented in your assignment. For each item, indicate what file/line number is testing it. (Document Section)
	100%	1	1		2	Document Section 2 - Design description/rationale (Section 2-i and 2-ii)
	100%	2	2	A	2.1	Description/rationale of the overall structure of the solution and the roles of the individual components used in the applied solution. (Document Section 2—i)
	100%	2	2	A	2.2	Description of the purpose of each semantic checking phase involved in the implementation. For each phase, mapping of semantic actions to AST nodes, along with a description of the effect/role of each semantic action (document Section 2-ii).
	100%				3	Document Section 3 - Description of tools/libraries/techniques used in the analysis/implementation. Description of other tools that might have been used. Justification of why the chosen tools were selected
	100%	1	1	A	3.1	All tools used in the analysis/implementation are mentioned.
	100%	1	1	A	3.2	Justification for all tools.
	100%	20	20		4	Correct implementation of a code generator
	100%	1	1	A	4.1	1.1 Allocate memory for basic types (integer, float).
	100%	0.5	0.5	A	4.2	1.2 Allocate memory for arrays of basic types.
	100%	0.5	0.5	A	4.3	1.3 Allocate memory for arrays of objects.
	100%	0.5	0.5	A	4.4	1.4 Allocate memory for arrays of objects.
	100%	1	1	A	4.5	2.1 Branch to a function's code block, execute the code block, branch back to the calling function.
	100%	1	1	A	4.6	2.2 Pass parameters as local values to the function's code block.
	100%	1	1	A	4.7	2.3 Upon execution of a return statement, pass the return value back to the calling function.
	100%	1	1	A	4.8	2.4 Call to member functions that can use their object's data members.
	100%	1.5	1.5	A	4.9	3.1 Assignment statement: assignment of the resulting value of an expression to a variable, independently of what is the expression to the right of the assignment operator.
	100%	1	1	A	4.10	3.2 Conditional statement: implementation of a branching mechanism.
	100%	1	1	A	4.11	3.3 Loop statement: implementation of a branching mechanism.
	100%	2	2	A	4.12	3.4 Input/Output statement: Move machine keyboard input/console output
	100%	1	1	A	4.13	5.1 For array of basic types (integer and float), access to an array's elements.
	100%	1	1	A	4.14	5.2 For array of objects, access to an array's element's data members.
	100%	1	1	A	4.15	5.3 For objects, access to members of basic types.
	100%	1	1	A	4.16	5.4 For objects, access to members of array or object types.
	100%	2	2	A	4.17	5.1 Computing the value of an entire complex expression.
	100%	1	1	A	4.18	5.2 Expression involving an array factor whose indexes are themselves expressions.
	100%	1	1	A	4.19	5.3 Expression involving an object factor referring to object members.
	100%	3	3		5	Output - .moon file. Output of correct executable code to a file
	100%	1	1	A	5.1	Clarity - generated code includes comments that enables to clearly identify/understand each portion of generated code
	100%	1	1	A	5.2	Correctness - the generated code is correct and executable
	100%	15	15		6	Test cases - completeness of testing (in addition to the provided files)
	100%	0.75	0.75	A	6.1	1.1 Allocate memory for basic types (integer, float).
	100%	0.375	0.375	A	6.2	1.2 Allocate memory for arrays of basic types.
	100%	0.375	0.375	A	6.3	1.3 Allocate memory for arrays of objects.
	100%	0.5	0.5	A	6.4	1.4 Allocate memory for arrays of objects.
	100%	0.75	0.75	A	6.5	2.1 Branch to a function's code block, execute the code block, branch back to the calling function.
	100%	0.75	0.75	A	6.6	2.2 Pass parameters as local values to the function's code block.
	100%	0.75	0.75	A	6.7	2.3 Upon execution of a return statement, pass the return value back to the calling function.
	100%	0.75	0.75	A	6.8	2.4 Call to member functions that can use their object's data members.
	100%	1.125	1.125	A	6.9	3.1 Assignment statement: assignment of the resulting value of an expression to a variable, independently of what is the expression to the right of the assignment operator.
	100%	0.75	0.75	A	6.10	3.2 Conditional statement: implementation of a branching mechanism.
	100%	0.75	0.75	A	6.11	3.3 Loop statement: implementation of a branching mechanism.
	100%	1.5	1.5	A	6.12	3.4 Input/Output statement: Move machine keyboard input/console output
	100%	0.75	0.75	A	6.13	5.1 For array of basic types (integer and float), access to an array's elements.
	100%	0.75	0.75	A	6.14	5.2 For array of objects, access to an array's element's data members.
	100%	0.75	0.75	A	6.15	5.3 For objects, access to members of basic types.
	100%	0.75	0.75	A	6.16	5.4 For objects, access to members of array or object types.
	100%	1.5	1.5	A	6.17	5.1 Computing the value of an entire complex expression.
	100%	0.75	0.75	A	6.18	5.2 Expression involving an array factor whose indexes are themselves expressions.
	100%	0.75	0.75	A	6.19	5.3 Expression involving an object factor referring to object members.
	100%	4	4		7	Successful/correct use of tools/libraries/techniques in the analysis/implementation.
	100%	1	1	A	7.1	The program never crashes or throws exceptions.
	100%	2	2	A	7.2	Tools presented in the labs are used appropriately, or comparable tools are used appropriately

Grading rubric			weight	min	max
Document Section 1 - Checklist of code generation items implemented, along with mapping to test				2.00	
Document Section 2 - Design description/rationale (Section 2-i and 2-ii)				5.00	
Document Section 3 - Description of tools/libraries/techniques used in the analysis/implementation				2.00	
Correct implementation of a code generator				20.00	
Output - .moon file. Output of correct executable code to a file				3.00	
Test cases - completeness of testing (in addition to the provided files)				15.00	
Successful/correct use of tools/libraries/techniques in the analysis/implementation				4.00	
Total				50.00	