Marker:

Key marking		Language features as implemented and tested in the assignment
Automatic		
A	A	1.2. 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 objects.
A	A	1.4. 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	3.4 Loop statement: implementation of a looping mechanism.
A	A	3.5 Input/output statement: Mixer-machine keyboard-input/console-output
A	A	4.1. For objects, access to members of basic types, to an array's elements.
A	A	4.2. For arrays of objects, access to an array's element's data members.
A	A	4.3. For objects, access to members of basic types.
A	A	4.4. 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	Total %	grade	unweighted	letter		
	100%	2	2		**1 Document Section 1 - Checklist of code generation items implemented, along with mapping to test files.**	1 : .
	100%	2	2	A	1.1 Using file (1-3) devised-tot 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 Section1)	1.1 : A .
	100%	5	5		**2 Document Section 2 - Design description/rationale (Section 2-1 and 2-3)**	2 : .
	100%	1	1	A	2.1 Description of the overall architecture, including a diagram showing the main components and the individual components used in the applied solution. (Document Section 2 – 1)	2.1 : A .
	100%	1	1	A	2.2 Description of the-process for each phase, including-a-diagram-showing-the-main-steps-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 2-4)	2.2 : A .
	100%	2	2		**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**	3 : .
	100%	1	1	A	3.1 All tools used in the analysis/implementation are mentioned.	3.1 : A .
	100%	1	1	A	3.2 Justification for all tools.	3.2 : A .
	100%	20	20		**4 Correct implementation of a code generator**	4 : .
	100%	1	1	A	4.1 1.1. Allocate-memory for basic types (integer, float).	4.1 : A .
	100%	0.5	0.5	A	4.2 1.2. Allocate-memory for arrays of basic types.	4.2 : A .
	100%	0.5	0.5	A	4.3 1.3. Allocate-memory for objects.	4.3 : A .
	100%	0.5	0.5	A	4.4 1.4. Allocate-memory for arrays of objects.	4.4 : A .
	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.	4.5 : A .
	100%	1	1	A	4.6 2.2. Pass parameters as local values to the function's code-block.	4.6 : A .
	100%	1	1	A	4.7 2.3. Upon execution of a return statement, pass the return value back to the calling function.	4.7 : A .
	100%	1	1	A	4.8 2.4. Call to member functions that can-use their object's data members.	4.8 : A .
	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.	4.9 : A .
	100%	1	1	A	4.10 3.2 Conditional statement: implementation of a branching mechanism.	4.10 : A .
	100%	1	1	A	4.11 3.4 Loop statement: implementation of a looping mechanism.	4.11 : A .
	100%	2	2	A	4.12 3.5 Input/output statement: Mixer-machine keyboard-input/console-output	4.12 : A .
	100%	1	1	A	4.13 4.1. For objects, access to members of basic types (integer and float), access to an array's elements.	4.13 : A .
	100%	1	1	A	4.14 4.2. For arrays of objects, access to an array's element's data members.	4.14 : A .
	100%	1	1	A	4.15 4.3. For objects, access to members of basic types.	4.15 : A .
	100%	1	1	A	4.16 4.4. For objects, access to members of array or object types.	4.16 : A .
	100%	2	2	A	4.17 5.1. Computing the value of an entire complex expression.	4.17 : A .
	100%	1	1	A	4.18 5.2. Expression involving an array factor whose indexes are themselves expressions.	4.18 : A .
	100%	1	1	A	4.19 5.3. Expression involving an object factor referring to object members.	4.19 : A .
	100%	4	4		**5 Output – .mcom file: Output of correct executable code to a file**	5 : .
	100%	1	1	A	5.1 Clarity - error messages include a line number where the error was found	5.1 : A .
	100%	1	1	A	5.2 Clarity - error messages describe what specific kind of error was found	5.2 : A .
	100%	1	1	A	5.3 Consistency - error messages are consistent in their-wording when-a-similar error is not reported as errors	5.3 : A .
	100%	1	1	A	5.4 Consistency - all errors are reported - exact-parsing-when-error-recovery is triggered.	5.4 : A .
	100%	15	15		**6 Test cases - completeness of testing (in addition to the provided files)**	6 : .
	100%	0.75	0.75	A	6.1 1.1. Allocate-memory for basic types (integer, float).	6.1 : A .
	100%	0.375	0.375	A	6.2 1.2. Allocate-memory for arrays of basic types.	6.2 : A .
	100%	0.375	0.375	A	6.3 1.3. Allocate-memory for objects.	6.3 : A .
	100%	0.375	0.375	A	6.4 1.4. Allocate-memory for arrays of objects.	6.4 : A .
	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.	6.5 : A .
	100%	0.75	0.75	A	6.6 2.2. Pass parameters as local values to the function's code-block.	6.6 : A .
	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.	6.7 : A .
	100%	0.75	0.75	A	6.8 2.4. Call to member functions that can-use their object's data members.	6.8 : A .
	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.	6.9 : A .
	100%	0.75	0.75	A	6.10 3.2 Conditional statement: implementation of a branching mechanism.	6.10 : A .
	100%	0.75	0.75	A	6.11 3.4 Loop statement: implementation of a looping mechanism.	6.11 : A .
	100%	1.5	1.5	A	6.12 3.5 Input/output statement: Mixer-machine keyboard-input/console-output	6.12 : A .
	100%	0.75	0.75	A	6.13 4.1. For objects, access to members of basic types (integer and float), access to an array's elements.	6.13 : A .
	100%	0.75	0.75	A	6.14 4.2. For arrays of objects, access to an array's element's data members.	6.14 : A .
	100%	0.75	0.75	A	6.15 4.3. For objects, access to members of basic types.	6.15 : A .
	100%	0.75	0.75	A	6.16 4.4. For objects, access to members of array or object types.	6.16 : A .
	100%	1.5	1.5	A	6.17 5.1. Computing the value of an entire complex expression.	6.17 : A .
	100%	0.75	0.75	A	6.18 5.2. Expression involving an array factor whose indexes are themselves expressions.	6.18 : A .
	100%	0.75	0.75	A	6.19 5.3. Expression involving an object factor referring to object members.	6.19 : A .
	100%	2	2		**7 Successful/correct use of tools/libraries/techniques in the analysis/implementation.**	7 : .
	100%	1	1	A	7.1 The program never crashes or throws exceptions	7.1 : A .
	100%	2	2	A	7.2 Tools presented in the labs are used appropriately, or comparable tools are used appropriately	7.2 : A .

Marker instructions

Enter values only in the red cells. Everything else is calculated automatically.
In Column G enter either A, B, C, or F for each marking element
Enter notes in column A. These notes will be then automatically uploaded on moodle as feedback to the students.

Designer instructions

When you add lines, add them only in the middle of the area, or else the calculation is going to be wrong
To add lines, add a new row, then copy into it one of the existing rows.

Grading rules	table	grade	indicator
Document Section 1 - Checklist of code generation items implemented, along with mapping to test	1	2.00	1
Document Section 2 - Design description/rationale (Section 2-1 and 2-3)	2	5.00	2
Document Section 3 - Description of tools/libraries/techniques used in the analysis/implementation	3	2.00	3
Correct implementation of a code generator	4	20.00	4
Output – .mcom file: Output of correct executable code to a file	5	4.00	5
Test cases - completeness of testing (in addition to the provided files)	6	15.00	6
Successful/correct use of tools/libraries/techniques in the analysis/implementation	7	4.00	7
total	%	50.00	