

**Concordia University  
Department of Computer Science  
and Software Engineering**

**Compiler Design (COMP 442/6421)  
Winter 2013**

**Final Project Presentation Grading Sheet**

<b>Deadline:</b>	April 15-17, 2013
<b>Evaluation:</b>	30% of final grade
<b>Late submission:</b>	not accepted

**Instructions**

You must deliver an operational version demonstrating the integrated capacities of your compiler. This is about demonstrating that your project has been effectively aimed at solving specific project problems. The tasks involved in building a working compiler have been identified, listed, and attributed some individual marks. The objective of your presentation is to demonstrate by usage the extent to which your compiler is achieving the list of tasks.

During the presentation, you have to do an individual demonstration of each functional requirement as listed on the following grading sheet. For each functional requirement, you are expected to come prepared with at least one test case dedicated to its demonstration. You are thus also graded according to how effectively you can demonstrate that the listed features are implemented. Negative marking will be applied in cases of ineffectiveness of demonstration or lack of preparation, up to a maximum of -10%.

If you cannot really demonstrate the features through execution, you will have to prove that the features are implemented by explaining how your code implements the features, in which case you may be given some marks. Even in such cases, you have to demonstrate that you are well prepared for the presentation, and that you can easily provide clear explanations as questions are asked about the functioning of your code.

**Identification**

<b>Student Name</b>	<b>Evaluator Name</b>	<b>Evaluator Signature</b>	<b>Presentation Time</b>

## Evaluation criteria and grading scheme

	effective demo.	weight	mark
<b>Interface</b>		<b>5</b>	
input interface: user-provided file name	oo	2	
output interface: clarity of standard output, alternate output to different files	oo	3	
<b>Lexical analysis</b>		<b>10</b>	
follows assignment 1 specifications		1	
error detection: completeness	oo	1	
error messages: clarity		1	
error recovery: no cascades, does not halt		1	
output token stream: show output in file	oo	2	
integers and floating point numbers	oo	2	
comments: inline, block, unending	oo	2	
<b>Syntactic analysis</b>		<b>30</b>	
follows assignment 2 specifications		2	
error detection: completeness	oo	2	
error messages: clarity		2	
error recovery: no cascades, does not halt		2	
output derivation: show output in file	oo	2	
main function, free functions	oo	2	
variable declaration: int, float, class, array	oo	2	
complex expressions (with arithmetic, relational and logic operators)	oo	5	
conditional statement, including nested without brackets	oo	2	
loop statement, including nested without brackets	oo	2	
class declarations: data members, methods	oo	3	
access to class members	oo	2	
access to arrays : uni- and multi-dimensional	oo	2	
<b>Semantic analysis</b>		<b>30</b>	
follows assignment 3 specifications		2	
error detection: completeness	oo	2	
error messages: clarity		2	
output symbol tables: show output in file	oo	3	
attribute migration: explain in compiler code	oo	3	
undefined id: variable, class, function	oo	2	
undefined member: data member, method	oo	2	
circular references: multiple passes	oo	2	
multiply defined id: variable, class, function, class member	oo	2	
function calls: right number and type of parameters upon call	oo	3	
type checking: complex expression	oo	3	
type checking: assignment	oo	2	
type checking: return value	oo	1	
function without return statement	oo	1	
<b>Code generation</b>		<b>25</b>	
variable declaration: int & float	oo	1	
variable declaration: objects	oo	2	
variable declaration: arrays	oo	1	
function bodies: code block structure	oo	1	
loop statement: code block, jump-looping upon condition	oo	2	
conditional statement: code block, jumping on condition	oo	2	
read/write statements: read from keyboard, write to standard output	oo	1	
complex expressions: arithmetic, relational and logic operators	oo	3	
function call mechanism: jump on call, return value	oo	2	
parameter passing mechanism	oo	3	
passing array/object as parameter	oo	1	
recursive function calls	oo	1	
floating point numbers computation	oo	1	
class declaration	oo	1	
method calls	oo	1	
arrays processing (uni- and multi-dimensional)	oo	1	
arrays of objects	oo	1	