Concordia University
Department of Computer Science
and Software Engineering

Compiler Design (COMP 442/6421)
Winter 2014

Final Project Presentation Grading Sheet

Deadline:	April 14-15, 2014
Evaluation:	30% of final grade
Late submission:	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

Student Name	Evaluator Name	Evaluator Signature	Presentation Time

Evaluation criteria and grading scheme

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