Concordia University
Department of Computer Science
and Software Engineering

Compiler Design (COMP 442/6421)
Winter 2017

Final Project Presentation Grading Sheet

Deadline:	April 8-11, 2017
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.

The presentation also includes the evaluation of graduate attributes. For each attribute indicator listed, you are given a letter grade. The letter-to-numeric grade correspondence is the following: A:100%, B:75%, C:50%, F:0%

Identification

Student Name	Evaluator Name	Evaluator Signature	Presentation Time

Evaluation criteria and grading scheme

	effectiveness	weight	mark
Interface		**2**	
input interface: user-provided file name	○○	1	
output interface: clarity of standard output, alternate output to different files	○○	1	
Lexical analysis		**8**	
lexical error detection and reporting: completeness and clarity	○○	2	
output token stream: show output in file	○○	2	
integers and floating point numbers processed according to original specifications	○○	2	
comments: inline, block, unending	○○	2	
Syntactic analysis		**27**	
syntactic error detection, reporting and recovery: completeness and clarity	○○	3	
output derivation: show output in file	○○	2	
main function, free functions	○○	2	
variable declarations: int, float, class, array	○○	2	
complex expressions (all arithmetic, relational and logic operators in one expression)	○○	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, including multiply nested and including arrays	○○	3	
access to arrays : uni- and multi-dimensional, using expressions as index	○○	3	
Semantic analysis		**28**	
semantic error detection and reporting: completeness and clarity	○○	2	
output symbol tables: show output in file	○○	3	
attribute migration mechanism: explain in compiler code	○○	3	
undefined id: variable, class, function	○○	2	
undefined member: data member, method, including deeply nested	○○	3	
forward/circular references: implementation of two passes	○○	2	
multiply defined id: variable, class, function, class member	○○	2	
arrays: using right number of dimensions	○○	2	
type checking of a complex expression	○○	3	
type checking of an assignment statement	○○	2	
type checking of the return value of a function	○○	2	
function calls: right number and types of parameters upon call	○○	2	
Code generation		**20**	
memory allocation: int and float variable declarations	○○	1	
memory allocation: array variable declarations	○○	1	
memory allocation: object variable declarations	○○	1	
loop statement: code block, jump, looping upon condition	○○	2	
conditional statement: code blocks, jumping on condition	○○	2	
Input/output: read from keyboard, write to standard output	○○	2	
expressions: arithmetic, relational and logic operators	○○	2	
expressions: composite expressions and intermediate results	○○	2	
function declaration code block (alias to jump to, jump back)	○○	1	
function call mechanism: jump on call, return value	○○	2	
parameter passing mechanism	○○	2	
offset calculation mechanism: arrays processing (uni- and multi-dimensional), using data members	○○	2	
Bonus marks:		**5**	
passing array/object as parameter	○○	1	
function call stack implementation: recursive/circular function calls	○○	2	
floating point numbers computation	○○	1	
method calls	○○	1	
arrays of objects	○○	1	
Functional Requirements —Total		**85**	

Graduate attributes		**letter**	**15**	
Attribute 1: Knowledge-base	Indicator 1.2: Show competence in tackling advanced engineering problems: Demonstrate understanding of the theoretical basis of the implementation.		2	
Attribute 2: Problem analysis	Indicator 2.1: Problem identification and formulation: Demonstrate that the implementation follows the original specifications. Demonstrate that the problem is clearly and completely understood,		2	
	Indicator 2.2: Modeling: Explain what models were used to analyze and implement the lexical/syntactical/semantic specifications.		2	
Attribute 4: Design	Indicator 4.1: Problem identification and information gathering: Demonstrate that the solution is well-adapted to the problem, and that unstated parts of the problem were uncovered as part of the development process.		2	
	Indicator 4.3: Architectural and detailed design: Description of the rationale and structure of the architectural design and detailed design justified against project requirements/constraints.		2	
Attribute 5: Use of Engineering tools	Indicator 5.2: Ability to evaluate and select appropriate tools: Justified adoption of tools in the project (e.g. programming language, compiler, IDE, libraries, project management tools, grammar analysis tools, etc).		1	
	Indicator 5.3: Ability to use tools: Proficient use of particular tools for the analysis and implementation.		2	
Attribute 7: Communication skills	Indicator 7.4: Oral presentation: Structure and demonstrated preparation of presentation, using appropriate presentation techniques. Demonstrated knowledge of code base/clarity of explanations.		2	
	Graduate Attributes — Total		**15**	

Total		**100**	