**CSc 201: Programming Language Principles**

**California State University, Sacramento**

**Fall Semester 2025**

**Section 01**

**Team 1**

**Team members:**

John Spaugh

Taro Kumagai

Niravkumar Tandel (nickname: Nirav)

**Project 1 Given on Tuesday of the 6th Week 9/30/2025**

**Submission Deadline: Wednesday of the 10th Week 10/29/2025**

**Team Presentation: Thursday of the 10th Week 10/30/2025**

**20 Points**

# General Information

This project is designed for you to write, using ML, an executable specification of an abstract language system for a simple imperative programming language. In such a system, the meaning of programs in this simple imperative language defined can be computed. Therefore, your executable specification of this language can serve as a rapid prototype of the implementation for this imperative language. This team project provides an opportunity of applying to practice the course materials on abstract syntax, denotational semantics, and functional programming language SML.

Before the start of your project, read carefully all sections of this document including **Project**

**Description**, **Project** **Requirements**, **Project Submissions**, and **the Abstract Grammar in BNF for a Graal-like Language for Project1**.As you work on your project, use them to guide you and follow the steps specified. When you progress at the specified steps, it is important to review the lectures on ML, abstract syntax, static semantics, and dynamic semantics.As an individual student, you should review, understand and contribute to each step and each part of this team project including the code written by your team.

# Project Description

**Step 1 Abstract Syntax 9/30/2025-10/7/2025**

**Submission deadline: 8pm of Tuesday of the 7th week 10/7/2025. 5 points total: 4 points for team submission, 1 point for individual attendance of the lecture on Thursday 10/2 mainly on abstract syntax, denotational semantics, detailed instruction for Project1 Step1. The team member attendance is the base of the team performance.**

* Understand fully the algorithm of the one sample program in a Graal-like language assigned to your team by executing and testing it in an imperative programming language environment which you are familiar with.
* In your ML file, use ML data type definition mechanisms to define an abstract syntax for a Graal-like language defined in this file and discussed in class. You need to use data constructors, type constructors, and recursions in your data type definitions.
* Include the given sample program in your ML file using comments. Then, step by step, using your data type definitions in ML for the abstract syntax, represent this sample program as an instance of data type Program you define (i.e., manual parse this sample program based on the abstract syntax you define.)
* NOTE: This step does not involve any function definitions.

**Step 2 Static Semantics 10/7/2025-10/17/2025**

**Submission deadline: 8pm of Friday of the 8th week 10/17/2025. 7 points total: 5 points for team submission, 2 points for individual attendance of lectures on 10/7 and 10/9 mainly on static semantics and detailed instruction for Project1 Step2. The team member attendance is the base of the team performance.**

* For the Graal-like language, define the domains for static semantics and associated operations/functions for the contextual information.
* Define the functions for static semantics. The function definitions are abstract syntax directed.
* Test your functions for static semantics by using proper pieces of your sample program.
* Apply the top function for static semantics to your sample program for the validity check.

**Step 3 Dynamic Semantics 10/17/2025-10/29/2025**

**Submission deadline: 8pm of Wednesday of the 10th week 10/29/2025. Each team is required to give a demo during class time on Thursday 10/30/2025. 8 points total: 6 points for team submission and demo, 2 points for individual attendance of lectures on 10/14 and 10/16 mainly on dynamic semantics and detailed instruction for Project1 Step3. The team member attendance is the base of the team performance.**

* Define the domain for dynamic semantics and associated operations/functions.
* Define functions for dynamic semantics. These function definitions are also abstract syntax directed.
* Test your functions for dynamic semantics by using proper pieces of your sample program.
* Define a top-level function that allows your functions for static semantics and your functions for dynamic semantics work together. Apply the top function to your sample program with an appropriate definition of the initial program state, then check the result.

# Requirements for the Organization/Sections of SML Definitions/Code in Your One .ml (or .sml) File

**Step1:** Definitions for the abstract syntax and representation of a sample program

* type/datatype definitions
* comments to include the Graal-like sample program assigned to your team
* representation of the Graal-like sample program assigned to your team **Step2:** Definitions for the static semantics
* type/datatype definitions
* function definitions, and testing in the between **Step3:** Definitions for the dynamic semantics
* type/datatype definitions
* function definitions, and testing in the between

# Requirements for Your Project Work

* You are required to attend lectures of Th 10/2, T 10/7, Th 10/9, T 10/14, and Th 10/16 that are on the Project1 materials and then contribute to your team project by directly applying the materials to your project work.
* You are required to submit your team project work on time. There is a **firm deadline** for each step. After the firm deadline, zero credit will be given to your team for the project work of that step.
* You are required to submit your complete work and SML accepted work for each step. Zero credit will be given to your project work of that step if there is any incompletion.
* As a teaching project, it is not a black box project at all. **You are required to follow all specific and also detailed instructions and requirements given in lectures for this project of this semester Fall 2025.** Zero credit will be given to your project work if any part (e.g., definition, testing) of your project work does not follow the given specific and also detailed instructions and requirements, if any part of your project work is not what is required for Fall 2025, and/or if there is any violation of academic honesty. Please review the University’s Policies on Academic Honesty.
* You are required to define rule-based, pattern matching-based recursive functions. Before you define each function, use SML comments to clearly specify the function signature.
* You are required to test functions as well as the entire program/specification as instructed during lectures. For each testing case you use to test a function, use SML comments to clearly specify what the testing case is.
* You are required to ensure the readability of your SML code/program layout. You need to use SML comments to clearly indicate where each step of this project starts.

# Requirements for Project Submissions Attention: Submissions are via email not via Canvas

At the end of each step before the deadline, each team should **send one email only** to the instructor at zhangc@csus.edu as your team project submission. You are required to have the **subject line: CSC201 Sec# Proj1 Team# Step#**. Email communication with the instructor for the project must include all team members.

When you submit your work of one step, your work in the previous step(s) needs to be included in your same file submission.You are required to submit the following for each step of your team work:

1. Your executable .ml (or .sml) file, i.e., a plain .ml.txt (or .sml.txt) file. This file needs to include the following as SML comments at the top: CSC 201 Section #, Team #, full

names of all team members. The file name should be

**Sec#Proj1Team#Step#.ml.txt (or** **Sec#Proj1Team#Step#.sml.txt).**

1. Your result file generated by SML, a plain .txt file. The file name should be **Sec#Proj1Team#Step#Result.txt**.
2. Your .docx file including all your executable ml code and comments followed by the result generated by SML. This file can be created by copying the content of BOTH your files 1) and 2).

At the top of this .docx file, please also list the following: CSC 201 Section #, Team #,

full names of all team members. The file name should be **Sec#Proj1Team#Step#Whole.docx**.

**The Abstract Grammar of Graal from B. Meyer’s Introduction to the Theory of Programming Languages** (see Meyer’s book)

# The Abstract Grammar of a Graal-like Language in BNF for Project1

<Program> ::= <DeclarationList> <Instruction>

<DeclarationList> ::= <Declaration> \*

<Declaration> ::= <Variable> <Type>

<Type> ::= TypeName1Bool | TypeName2Int

<Instruction> ::= Skip |

<Variable> <Expression> |

<Expression> <Instruction> <Instruction> |

<Expression> <Instruction> |

<Instruction> \*

<Expression> ::= <Variable> |

<Integer\_Constant> |

<Boolean\_Constant> |

<Expression> <Expression> <Operator>

<Operator> ::= <Arithmatic\_Op> | <Relational\_Op> | <Boolean\_Op>

<Boolean\_Op> ::= And | Or

<Relational\_Op> ::= Lt | Le | Eq | Ne | Ge | Gt

<Arithmatic\_Op> ::= Plus | Minus | Times | Div

<Variable> ::= S (String)

<Boolean\_Constant> ::= B (Bllolean)

<Integer\_Constant> ::= Z (integer)