**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**

# 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.

Below is Sec1Proj1Team1Step1.ml

(\* CSC201, Section 1, TEAM 1, John Spaugh,

Taro Kumagai,

Niravkumar Tandel \*)

(\*---- Sec1Proj1Team1Step1 ----------------------\*)

(\* Abastract Grammar of a Graal-like Language in BNF for Project \*)

type Integer\_Constant = int;

type Boolean\_Constant = bool;

datatype Variable = S of string;

datatype Arithmatic\_Op = Plus | Minus | Times | Div;

datatype Relational\_Op = Lt | Le | Eq | Ne | Ge | Gt;

datatype Boolean\_Op = And | Or;

datatype Operator = AOp of Arithmatic\_Op | ROp of Relational\_Op | BOp of Boolean\_Op;

datatype Expression = Var of Variable |

IC of Integer\_Constant |

BC of Boolean\_Constant |

EEO of Expression \* Expression \* Operator;

datatype Instruction = Skip |

VE of(Variable \* Expression) |

IfThenElse of(Expression \* Instruction \* Instruction) |

WhileLoop of(Expression \* Instruction) |

Seq of Instruction list;

datatype Type = TypeName1Bool | TypeName2Int ;

type Declaration = Variable \* Type;

type DeclarationList = Declaration list;

type Program = DeclarationList \* Instruction;

(\* -------------------

cpp file

#include <cstdlib>

#include <iomanip>

#include <iostream>

int main(){ //int argc, char \*argv[]){

int n;

int cur;

int prev1;

int prev2;

int i;

int answer;

n=15;

if(n == 0){

answer =2;

}else{

i=1;

if(n ==1){

answer=1;

}else{

prev2 =2;

prev1 =1;

while (i < n){

i++; // i = i +1;

cur = prev1 + prev2;

prev2 = prev1;

prev1 = cur;

}

answer = cur;

}

}

std::cout << "Answer: " << answer << std::endl;

return 0;

}

---------------------------\*)

Below is Sec1Proj1Team1Step1Result.txt

Standard ML of New Jersey [Version 110.99.8; 32-bit; April 25, 2025]

- type Integer\_Constant = int

type Boolean\_Constant = bool

datatype Variable = S of string

datatype Arithmatic\_Op = Div | Minus | Plus | Times

datatype Relational\_Op = Eq | Ge | Gt | Le | Lt | Ne

datatype Boolean\_Op = And | Or

datatype Operator

= AOp of Arithmatic\_Op

| BOp of Boolean\_Op

| ROp of Relational\_Op

datatype Expression

= BC of bool

| EEO of Expression \* Expression \* Operator

| IC of int

| Var of Variable

datatype Instruction

= IfThenElse of Expression \* Instruction \* Instruction

| Seq of Instruction list

| Skip

| VE of Variable \* Expression

| WhileLoop of Expression \* Instruction

datatype Type = TypeName1Bool | TypeName2Int

type Declaration = Variable \* Type

type DeclarationList = Declaration list

type Program = DeclarationList \* Instruction