(\* 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 = TypeBool | TypeInt ;

type Declaration = Variable \* Type;

type DeclarationList = Declaration list;

type Program = DeclarationList \* Instruction;

(\* Val Variable Declarations \*)

val var\_n = S "n"

val var\_cur = S "cur"

val var\_prev1 = S "prev1"

val var\_prev2 = S "prev2"

val var\_i = S "i"

val var\_answer = S "answer"

val var\_temp = S "temp"

val declaration\_n = (var\_n, TypeInt)

val declaration\_cur = (var\_cur, TypeInt)

val declaration\_prev1 = (var\_prev1, TypeInt)

val declaration\_prev2 = (var\_prev2, TypeInt)

val declaration\_i = (var\_i, TypeInt)

val declaration\_answer = (var\_answer, TypeInt)

(\* Full Declaration List \*)

val allDeclarations = [

declaration\_n,

declaration\_cur,

declaration\_prev1,

declaration\_prev2,

declaration\_i,

declaration\_answer

]

(\* Relational Logic \*)

val n\_equals\_0 = EEO (Var var\_n, IC 0, ROp Eq) (\* n == 0 \*)

val n\_equals\_1 = EEO (Var var\_n, IC 1, ROp Eq) (\* n == 1 \*)

val i\_lessthan\_n = EEO (Var var\_i, Var var\_n, ROp Lt) (\* i < n \*)

(\* Assignment Instructions, FORMAT assign\_variable\_value \*)

val assign\_n\_15 = VE (var\_n, IC 15) (\* n = 15 \*)

val assign\_answer\_1 = VE (var\_answer, IC 1) (\* answer = 1 \*)

val assign\_answer\_2 = VE (var\_answer, IC 2) (\* answer = 2 \*)

val assign\_i\_1 = VE (var\_i, IC 1) (\* i = 1 \*)

val assign\_prev1\_1 = VE (var\_prev1, IC 1) (\* prev1 = 1 \*)

val assign\_prev2\_2 = VE (var\_prev2, IC 2) (\* prev2 = 2 \*)

val assign\_answer\_cur = VE (var\_answer, Var var\_cur) (\* answer = cur \*)

val assign\_prev2\_prev1 = VE (var\_prev2, Var var\_prev1) (\* prev2 = prev1 \*)

val assign\_prev1\_cur = VE (var\_prev1, Var var\_cur) (\* prev1 = cur\*)

(\* Arithmatic Expressions \*)

val add\_i\_1 = EEO (Var var\_i, IC 1, AOp Plus) (\* i + 1 \*)

val add\_prev1\_prev2 = EEO (Var var\_prev1, Var var\_prev2, AOp Plus) (\* prev1 + prev2 \*)

(\* Arithmatic Instructions \*)

val ipp = VE (var\_i, add\_i\_1) (\* i++ \*)

val cur\_prev1\_plus\_prev2 = VE (var\_cur, add\_prev1\_prev2) (\* cur = prev1 + prev2 \*)

(\* Instructions \*)

val insideWhile = [

ipp,

cur\_prev1\_plus\_prev2,

assign\_prev2\_prev1,

assign\_prev1\_cur

]

val whileLoop = WhileLoop(i\_lessthan\_n, Seq insideWhile)

val inner\_Else = [

assign\_prev2\_2,

assign\_prev1\_1,

whileLoop,

assign\_answer\_cur

]

val inner\_ifThenElse = IfThenElse(n\_equals\_1, assign\_answer\_1, Seq inner\_Else)

val outer\_Else = [

assign\_i\_1,

inner\_ifThenElse

]

val outer\_ifThenElse = IfThenElse(n\_equals\_0, assign\_answer\_2, Seq outer\_Else)

val allInstructions = [assign\_n\_15, outer\_ifThenElse]

(\* The Program LUCAS\*)

val lucas = (allDeclarations, Seq allInstructions)

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

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;

}

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

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 = TypeBool | TypeInt

type Declaration = Variable \* Type

type DeclarationList = Declaration list

type Program = DeclarationList \* Instruction

val var\_n = S "n" : Variable

val var\_cur = S "cur" : Variable

val var\_prev1 = S "prev1" : Variable

val var\_prev2 = S "prev2" : Variable

val var\_i = S "i" : Variable

val var\_answer = S "answer" : Variable

val var\_temp = S "temp" : Variable

val declaration\_n = (S "n",TypeInt) : Variable \* Type

val declaration\_cur = (S "cur",TypeInt) : Variable \* Type

val declaration\_prev1 = (S "prev1",TypeInt) : Variable \* Type

val declaration\_prev2 = (S "prev2",TypeInt) : Variable \* Type

val declaration\_i = (S "i",TypeInt) : Variable \* Type

val declaration\_answer = (S "answer",TypeInt) : Variable \* Type

val allDeclarations =

[(S "n",TypeInt),(S "cur",TypeInt),(S "prev1",TypeInt),(S "prev2",TypeInt),

(S "i",TypeInt),(S "answer",TypeInt)] : (Variable \* Type) list

val n\_equals\_0 = EEO (Var (S "n"),IC 0,ROp Eq) : Expression

val n\_equals\_1 = EEO (Var (S "n"),IC 1,ROp Eq) : Expression

val i\_lessthan\_n = EEO (Var (S "i"),Var (S "n"),ROp Lt) : Expression

val assign\_n\_15 = VE (S "n",IC 15) : Instruction

val assign\_answer\_1 = VE (S "answer",IC 1) : Instruction

val assign\_answer\_2 = VE (S "answer",IC 2) : Instruction

val assign\_i\_1 = VE (S "i",IC 1) : Instruction

val assign\_prev1\_1 = VE (S "prev1",IC 1) : Instruction

val assign\_prev2\_2 = VE (S "prev2",IC 2) : Instruction

val assign\_answer\_cur = VE (S "answer",Var (S "cur")) : Instruction

val assign\_prev2\_prev1 = VE (S "prev2",Var (S "prev1")) : Instruction

val assign\_prev1\_cur = VE (S "prev1",Var (S "cur")) : Instruction

val add\_i\_1 = EEO (Var (S "i"),IC 1,AOp Plus) : Expression

val add\_prev1\_prev2 = EEO (Var (S "prev1"),Var (S "prev2"),AOp Plus)

: Expression

val ipp = VE (S "i",EEO (Var (S "i"),IC 1,AOp Plus)) : Instruction

val cur\_prev1\_plus\_prev2 =

VE (S "cur",EEO (Var (S "prev1"),Var (S "prev2"),AOp Plus)) : Instruction

val insideWhile =

[VE (S "i",EEO (Var (S "i"),IC 1,AOp Plus)),

VE (S "cur",EEO (Var (S "prev1"),Var (S "prev2"),AOp Plus)),

VE (S "prev2",Var (S "prev1")),VE (S "prev1",Var (S "cur"))]

: Instruction list

val whileLoop =

WhileLoop

(EEO (Var (S "i"),Var (S "n"),ROp Lt),

Seq

[VE (S "i",EEO (Var (S #),IC 1,AOp Plus)),

VE (S "cur",EEO (Var (S #),Var (S #),AOp Plus)),

VE (S "prev2",Var (S "prev1")),VE (S "prev1",Var (S "cur"))])

: Instruction

val inner\_Else =

[VE (S "prev2",IC 2),VE (S "prev1",IC 1),

WhileLoop

(EEO (Var (S "i"),Var (S "n"),ROp Lt),

Seq

[VE (S "i",EEO (Var #,IC #,AOp #)),

VE (S "cur",EEO (Var #,Var #,AOp #)),VE (S "prev2",Var (S "prev1")),

VE (S "prev1",Var (S "cur"))]),VE (S "answer",Var (S "cur"))]

: Instruction list

val inner\_ifThenElse =

IfThenElse

(EEO (Var (S "n"),IC 1,ROp Eq),VE (S "answer",IC 1),

Seq

[VE (S "prev2",IC 2),VE (S "prev1",IC 1),

WhileLoop

(EEO (Var (S #),Var (S #),ROp Lt),

Seq [VE (#,#),VE (#,#),VE (#,#),VE (#,#)]),

VE (S "answer",Var (S "cur"))]) : Instruction

val outer\_Else =

[VE (S "i",IC 1),

IfThenElse

(EEO (Var (S "n"),IC 1,ROp Eq),VE (S "answer",IC 1),

Seq

[VE (S "prev2",IC 2),VE (S "prev1",IC 1),

WhileLoop (EEO (Var #,Var #,ROp #),Seq [VE #,VE #,VE #,VE #]),

VE (S "answer",Var (S "cur"))])] : Instruction list

val outer\_ifThenElse =

IfThenElse

(EEO (Var (S "n"),IC 0,ROp Eq),VE (S "answer",IC 2),

Seq

[VE (S "i",IC 1),

IfThenElse

(EEO (Var (S #),IC 1,ROp Eq),VE (S "answer",IC 1),

Seq [VE (#,#),VE (#,#),WhileLoop (#,#),VE (#,#)])]) : Instruction

val allInstructions =

[VE (S "n",IC 15),

IfThenElse

(EEO (Var (S "n"),IC 0,ROp Eq),VE (S "answer",IC 2),

Seq

[VE (S "i",IC 1),

IfThenElse

(EEO (Var #,IC #,ROp #),VE (S #,IC #),

Seq [VE #,VE #,WhileLoop #,VE #])])] : Instruction list

val lucas =

([(S "n",TypeInt),(S "cur",TypeInt),(S "prev1",TypeInt),(S "prev2",TypeInt),

(S "i",TypeInt),(S "answer",TypeInt)],

Seq

[VE (S "n",IC 15),

IfThenElse

(EEO (Var (S "n"),IC 0,ROp Eq),VE (S "answer",IC 2),

Seq [VE (S #,IC #),IfThenElse (EEO #,VE #,Seq #)])])

: (Variable \* Type) list \* Instruction