**CSE 351**

**Programming Languages**

**Homework Assignment #3**

**Due Date: 17th of April 2020 @17:00**

**1. A BNF grammar for expressions is given below:**

**<program>** **<type\_decls> <stmts>**

**<type\_decls>** **<type\_decl> <type\_decls>**

**<type\_decls>** **<type\_decl>**

**<type\_decl>** **<type> <var\_list> ;**

**Semantic Rule: <var\_list>.actual\_type 🡨 <type>.actual\_type**

**<var\_list>[1] 🡪 <var> , <var\_list>[2]**

**Semantic Rule: <var\_list>[2].actual\_type 🡨 <var\_list>[1].actual\_type**

**Semantic Rule: insert(<var>.string,<var\_list>[1].actual\_type)**

**/\* insert function updates the type of a variable on the Symbol Table. \*/**

**<var\_list> 🡪 <var>**

**Semantic Rule: insert(<var>.string,<var\_list>.actual\_type)**

**<type> 🡪 int**

**Semantic Rule: <type>.actual\_type 🡨 int**

**<type> 🡪 float**

**Semantic Rule: <type>.actual\_type 🡨 float**

**<stmts>** **<stmt> <stmts>**

**<stmts>** **<stmt>**

**<stmt>** **<var> = <expr>**

**Predicate: <var>.actual\_type == <expr>.actual\_type**

**<expr>[1]** **<var> + <expr>[2]**

**Semantic Rule: <expr>[1].actual\_type 🡨 if ((<var>.actual\_type == int) and**

**(<expr>[2].actual\_type == int))**

**int**

**else**

**float**

**endif**

**<expr>** **<var>**

**Semantic Rule: <expr>.actual\_type 🡨 <var>.actual\_type**

**<var> 🡪 A | B | C | D | E**

**Semantic Rule: <var>.actual\_type 🡨 lookup(<var>.string)**

**/\* lookup function returns the type from the Symbol Table of a given variable name \*/**

**a) Draw the parse tree for the following small program.**

**int A, B;**

**float D;**

**A = B + D ;**

**b) Show the flow of attributes in the parse tree you draw for part a**

Actualtype

Lookup()

Actualtype

int

Actualtype

int

**c) Indicate if any semantic error is found in the program.**

A = B + D ;

There is no ‘;’ in **<stmt>** **<var> = <expr>**

**2. While coding, it is always hard to match opening and closing curly braces (i.e. { and } symbols) in C and Java. So, we would like to write a unique number next to each curly brace pair, as shown in the example code below.**

public class Displayer { [1]

public static void main(String args[]) { [2]

System.out.println("You'll love Java!");

} [2]

} [1]

**Add semantic functions or predicates to achieve this for the given curly braces grammar below.**

**<curly>** **<curly> <curly>**

**<curly>[1]🡪 {<curly>[2]}**

**<curly> 🡪 { }**

**3. Add an attribute grammar over the given binary string grammar below, so that we can calculate its decimal value on the root node of any parse tree.**

**<binary> 🡪 <binary> <digit>**

**Semantic Rule: insert(<digit>.value)**

**/\* insert function insert digits to binary list\*/**

**<binary> 🡪 <digit>**

**Semantic Rule: insert(<digit>.value)**

**<digit> 🡪 0**

**Semantic Rule: <digit>.value 🡨 0**

**<digit> 🡪 1**

**Semantic Rule: <digit>.value 🡨 1**

**Draw the parse tree for the 1001 input, and show the attribute flow to calculate its decimal value of 9.**

.value

1