Compiler Theory CS447b: Quiz 3. **UWO, April 4, 2003.**

Student ID number:

Guidelines. The quiz is closed book and all notes are forbidden. The duration of the quiz is 30 minutes. All answers should be written in the *answer boxes*. No justifications for the answers are needed. You are expected to do this quiz on your own without assistance from anyone else in the class. If possible, please avoid pencils and use pens with dark ink. Thank you.

Exercise 1. We consider an abstract stack machine supporting arithmetic operations and the following instructions (exactly as the one of the course).

`push` v which pushes v onto the stack.

`rvalue` ℓ which pushes the **content** of **data location** ℓ onto the stack.

`lvalue` ℓ which pushes the **address** of **data location** ℓ onto the stack .

`pop` which throws away the value on top of the stack.

:= which places the R-value on top in the L-value below it, and both are popped.

`copy` which pushes a copy of the top value on the stack.

Translate the following C-like statement for this abstract stack machine.

`x = (a + b) * (c + d)`

Answer 1

lvalue x
rvalue a
rvalue b
+
rvalue c
rvalue d
+
*
:=

Exercise 2. Consider the following syntax-directed definition.

Production	Semantic Rule
$S \longmapsto E$	$E.env$:= initialEnv()
$E_1 \longmapsto$ **let id** $= E_2$ **in** E_3 **end**	$E_2.env$:= $E_1.env$
	$E_3.env$:= Update($E_1.env$, **id**, $E_2.val$)
	$E_1.val$:= $E_3.val$
$E_1 \longmapsto (E_2 + E_3)$	$E_1.val$:= $E_2.val + E_3.val$
	$E_2.env$:= $E_1.env$
	$E_3.env$:= $E_1.env$
$E \longmapsto$ **id**	$E.val$:= LookUp(**id**, $E.env$)
$E \longmapsto$ **1**	$E.val$:= **1**
$E \longmapsto$ **0**	$E.val$:= **0**

Write a translation scheme for this syntax-directed definition.

Answer 2

S	$\longmapsto$		{ $E.env$:= initialEnv() }
		E	
E_1	$\longmapsto$		{ $E_2.env$:= $E_1.env$ }
		let id $= E_2$	{ $E_3.env$:= Update($E_1.env$, **id**, $E_2.val$) }
		in E_3 **end**	{ $E_1.val$:= $E_3.val$ }
E_1	$\longmapsto$		{ $E_2.env$:= $E_1.env$ }
			{ $E_3.env$:= $E_1.env$ }
		$(E_2 + E_3)$	{ $E_1.val$:= $E_2.val + E_3.val$ }
E	$\longmapsto$	**id**	{ $E.val$:= LookUp(**id**, $E.env$) }
E	$\longmapsto$	**1**	{ $E.val$:= **1** }
E	$\longmapsto$	**0**	{ $E.val$:= **0** }

Exercise 3. Translate the following MOOL piece of code into three-address code.

```
while not (a = b) repeat {
    if (a < b) then {
       b := b - a;
   } else {
       a := a - b;
   };
};
```

The three-address code instructions you may need are of one of the forms:

- `goto L` where `L` is a symbolic label (like `L1, L2, ...`)
- `if x relop y goto L` where `relop` can be = or <
- `x:= y op z` where `op` can be −

Moreover your solution does **not** need to be obtained by one of the algorithms of the course. Just consider that you are programming a very basic calculator.

Answer 3

```
L1  if a = b goto L5
L2  if b < a goto L3
    b := b - a
    goto L4
L3  a := a - b
L4  goto L1
L5
```