11.6 Semantic analysis and code generation

Name: _____ Class: _____ Date: _____ **Total: 24 marks**

KEY WORDS symbol table • intermediate representation

Objective

Decorate trees with types and emit three-address IR.

1 Key knowledge and worked examples

■ The stage

Symbol table binds (name, type, scope); type rules walk the tree; the output is three-address IR ✓.

■ Worked example: the course program

`count = a1 + b2 * 3` (all int): `t1 = b2 * 3; t2 = a1 + t1; count = t2` ✓. Folding is literal-only ✓.

2 Practice

2.1 What does the symbol table map, and when is it built? [3]

2.2 IR for `x = 2 * 3` after folding – how many instructions? [3]

2.3 Why an IR at all (the arithmetic argument)? [3]

3 Exam-style questions

3.1 Program: `r = a + b * c - d / e` with all variables int.

Emit the three-address IR (temps numbered from t1). Give the instruction count and temp count. Give the type-check sequence (bottom-up, one line per operator). State which instruction changes if `d` were type str, and the error's reported position. [8]

3.2 Two programs:

(1) `y = 2 * 3 + x` (2) `y = x * 3 + 2`

For each: the folded-or-not IR and the reason. State the folding rule in one sentence. Then argue why the IR makes `m` languages compile to `n` targets with `m + n` components – name the alternative's cost ✓. [7]