

Unofficial homework problem

This homework problem was given in class for you to do on your own; don't turn it in. This problem also serves as a sample final exam problem: we have covered code generation in a different way than we have in recent semesters in this class, so this problem is a more representative code generation problem than the ones on the practice exams. See blurb in the sample exam section of the web page for more info on these differences.

Problem: Write a C++ member function for generating 3-address code for a C **if-else** statement from an AST representation of the code. The AST has two possible forms, depending on whether the optional else is present: your routine must be able to generate the correct code for both forms. Here are the parts of an `IfElse` statement:

```
class IfElse : public Statement {
protected:
    Expression *cond;
    Statement *thenPart;
    Statement *optElse;    // non-NULL iff there is an else
public:
    virtual void codeGen();    // you write this function
    . . .
}
```

Assumptions about generating code:

- you may use function `newTemp()` to generate unique temporary names (returns a string)
- you may use the function `newLabel()` to generate unique labels (returns a string).
- There is a polymorphic `genCode` function defined for all AST node types. For all `Expression` node types, `genCode` updates a place field, which stores the name which will hold its value. The place field can be accessed via the member function `getPlace()`.
- Use function `emit`, which takes a variable number of arguments, to generate 3AC statements as we have done in class. Here's an example of its use, where the parts in double-quotes are literal, and `myTemp` is a string variable.

```
emit(myTemp, " := ", myTemp, "+ 27;");
```

- Syntax of the 3AC to use:

```
x := y           Copy y's value into x
x := y op z      Where op is one of +, -, *, /, %, "and", "or"
x := op y        Where op is one of -, "not"
goto L           jump to label L
if y relop z goto L      Conditional jump to label L, where relop
                        is one of <, >, =, !=, >=, <=
x := y[i]        set x to the value in the mem. loc. i bytes beyond y
x[i] := y        give the mem loc i bytes beyond x the value in y
```

where the operands can be names or integer constants (except those used as arrays), and the result must be a name. For logical operations, assume non-zero is true, and zero is false