CS 540 Programming Fall 2014 Assignment 3: Typechecking UnCool

Due: Sun, November 9 at end of day (midnight)

Creating this assignment

As with all assignments in this class, you will need to do this assignment in several steps. BE SURE the previous steps work before going on to the next step.

The first step is to use Lex/Yacc to build a parser for UnCool. The grammar (syntax and lexical elements) is in the specification and you want to include the entire language (although we will only be dealing with parts of the language at times). You will initially have lots of shift/reduce errors. These will mostly go away once you add precedence rules for the operators. They will not go away entirely however. If you can parse the input examples on blackboard, your parser is fine.

Declaration Checking

The UnCool specification describes what constitute scopes for this language. When a name is used in some part of the program, you need to check to ensure that the name is visible at that point. If a variable not visible in the current scope is used, **print out a descriptive error message, including the variable name and the associated line number. Continue processing - in other words, do not exit the program.** Duplicate declarations in a given scope must also be indicated.

Type checking

All of the following will be checked by your typechecking tool. If you detect any of these issues, print out a descriptive error message (which includes a line number) and continue parsing.

1. All input files must have a Main class with a main() function.
2. All identifiers will be defined and used correctly with respect to their definition and scope.
3. Duplicate identifier declarations in the same scope will be identified. Remember that you can reuse the same name for a method and attribute in a given class.
4. The operands for all operators must be checked and the correct type for the overall expression should be determined. For example, '+' requires two integers and returns type integer. The following table gives types of for the operators.

Operation	Operand1	Operand2 (if applicable)	Result type
+,-,*	Int	Int	Int
<,<=,>,>=	Int	Int	Bool

<>,=	Any	*Any (matches operand 1)	Bool
isVoid	Any		Int
~ (uniminus)	Int		Int
Not	Bool		Bool
<-	Any	*Any (matches operand 1)	Same as operands

5. UnCool statements have types associated with them.
    a. An if statement must have a Boolean predicate and the types of the two branches (then/else) must match.
    b. While statements must have Boolean predicate and the type of the statement is the type of the associated expression.
    c. A let statement creates a new scope with types. It returns the type of the expression in the body.
    d. A sequence of expressions returns the type of the last expression in the sequence.
6. Arrays can only be dereferenced with integer indices.
7. For method definitions, the type of the method given must match the type of the body.
8. For method calls, the correct number of parameters must be used (although you do not need to check the types of the parameters for this assignment). The call returns the type defined in the definition. Recursion gets a little tricky when trying to check this so you are allowed to assume that a recursive call is always correct with respect to return type.

Submitting this assignment

Electronic submissions on blackboard. Be sure to include information in the comments

- on what language you used (C, C++ or Java),
- build instructions and
- the system (osf1, zeus) where you want me to build/test your assignment.

For this and all other assignments, but sure you give me ALL files needed to build your executable. The TA will not test your program on any executables you include.