Project 5. Not evaluating expressions

In directory /comp/150C++/project5/ are some source files that implement “unevaluated expressions,” together with a Makefile and some test programs. They provide a way for a programmer to write ordinary-looking expressions which, rather than being evaluated when execution flows through them, are preserved in a form that allows evaluation to occur later. This feature is sometimes called delayed evaluation, and could be used in conjunction with lazy evaluation or late binding heuristics. Unevaluated expressions have been provided by numerous programming languages, but they are not normally found in imperative languages. This project shows that they can be brought into C++, using the language’s extensibility features including polymorphism, operator overloading, and templates.

Class “Exp” represents an unevaluated expression. Objects representing integers, real numbers, and strings can be created, and they can be composed using ordinary unary and binary operators. Expressions can be stored and retrieved from simple “variables” which are identified by strings which act as the variables’ names. These strings need not be fixed; they can be generated dynamically by a subexpression. Operators are available for evaluating an Exp to yield an integer, real number, or string.

Two operators with special semantics are unimplemented in the given files. They are the unary address-of operator and the binary function-call operator. The address-of operator basically defeats the process of evaluation: when the evaluator encounters it in traversing the expression, it takes the operator’s subexpression in unevaluated form instead of evaluating it. The function call operator “f(a)” first evaluates the argument “a” and stores the result into a temporary place, and then evaluates the body “f” after arranging that, within “f,” the temporary place can be accessed as if it were a “variable” whose name is the empty string.

Examples of unevaluated expressions, and of the intended semantics of the unimplemented operators, will be given in sample files online and presented in class.

Your task is to provide the bodies of the “eval” methods for unary address-of and for binary function-call. You will also want to modify the dereference operator’s eval method so that it properly supports the new operators. When a value is retrieved from the symbol table, the result should be evaluated before being returned. (The given code does not do that.)

You can probably test and debug most effectively by writing your own mainlines, in addition to the ones provided.

Your program will be graded primarily for correctness of results, secondarily for lack of memory leaks, and possibly a bit for style. Extra credit will be given for discovering memory leaks that can fairly be blamed on dwc.

Submit your file by Saturday, May 1 at 5:00 p.m. using this command:

    provide 150C++ project5 exp2.cc