C/C++ 2015/16 programming exercise 3

This exercise is about the same problem as in Exercise 2, namely evaluating abstract syntax trees. However, you will now use virtual member functions in C++ rather than C trees and functions.

The class definitions are here:

`http://www.cs.bham.ac.uk/~hxt/2015/c-plus-plus/evalobj.h`

Your task is to implement the member functions `eval` of all derived classes. Your code should be in a file `evalobj.cpp`.

A minimal main file is here, though you may wish to write your own test cases:

`http://www.cs.bham.ac.uk/~hxt/2015/c-plus-plus/evalobjmain.cpp`

You may use the following includes:

```
#include <string>
using namespace std;
#include "evalobj.h"
```

To compile the code, you should use:

```
clang++ -std=c++11 -Werror -o evalobj evalobjmain.cpp evalobj.cpp
```

You can then test it with valgrind by using

```
valgrind -q ./evalobj
```

There should not be any memory *errors*, but memory *leaks* are OK in this exercise, as it is not about deallocation.

Of course, it would not be hard to implement destructors if you want to be tidy. Kleeneliness is next to Gödeliness.

This exercise counts for 5% of the module mark.

2 points are given if your `eval` functions work on tests without let bindings.

3 more points are given if your `eval` functions also work on tests with let bindings.