C/C++ 2017/18 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/2017/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`. The functions should not print anything.

You may use the following includes:

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

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

`http://www.cs.bham.ac.uk/~hxt/2017/c-plus-plus/evalobjmain.cpp`

To compile the code, you should use:

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

Your code must compile on the lab machines as above. If it fails to compile or produces any errors, it will be given 0 marks.

Marking scheme

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.

Valgrind should not report anything when used like this:

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

Given that deallocation is not a requirement, there is no flag for leak checking

```
--leak-check=full
```