Lab 6

Compute size/offset

For this lab, you will compute the size and offset of all the declarations and variable references. In addition, you will also compute the size of all decls and blocks.

Your implementation must conform to the following requirements:

1. The size and offset must be stored in instance variables with getters/setters. The getter must simply return the instance variable, it is not allowed to do any computation.
2. The instance variables must be declared in the lowest member of the class hierarchy where it makes sense to do so. In other words, if all (or most) of the descendants of a class need the feature, it can be declared/implemented in that class. As examples:
   1. All of the descendants of cDeclsNode use the size/offset, so the instance variables and getters/setters can be declared there.
   2. Most of the descendants of cStmtNode do NOT need size/offset, so the instance variables and getters/setters must not be visible to cStmtNode.
3. You must use the visitor pattern we discussed in class for your implementation.
4. You should override AttributesToString in each class that has a size and/or offset. AttributesToString must return a zero length string if both the size and offset are zero.
5. The size for any scope must reflect the high water mark for that scope.
6. You must reclaim and reuse the space for any inner scopes.
7. Your implementation must conform to good programming practices.

test0 is a concatenation of most of the tests from Lab 4 and Lab 5 with size/offset information added. Do not start with test0. Instead start with test1. When you complete all the other tests, test0 should pass.

When you start dealing with variable expressions that are arrays, you won’t be able to compute offsets. The reason is that, in general, offsets for array references can’t be computed until run-time. As a result, we need to handle them differently. The offset for an array reference should be the offset of the variable. The size should be the size of the varref. In addition to computing the size and offset, you also need to compute row sizes so that the final offset computation can be easily generated. Consider the following example:

array int[5] a1;

array int[7] a2;

int v0;

a1 v1; // size=20 offset=0

a2 v2; // size=140 offset=20

v1 = v2[1]; // size=20 offset=20 rowsizes=20

v0 = v2[1][2]; // size=4 offset=20 rowsizes=20,4

**Note on correctness criteria for this lab:**

There are different valid implementations of cFuncDeclNode from last week. These differences will result in differences in this week’s correct files. But these differences should NOT affect the size and offset computations. When I grade, if you solution comes up with the same size, offset, and rowsize values as mine, I will ignore other differences.