**Scott Vincent code review of Adam Bush, HW02**

|  |
| --- |
| It seems like this line: Node(int x, Node\* n, Node\* p) :: …  Is causing some big problems because as you can see in the debugger the N and P pointers are treated as Nodes and as such are creating a cascading list of interwoven pointers which are pointing to itself. |
|  |
|  |
| Address have been given to next and prev pointing to itself, the head node, but… |
|  |
|  |
|  |
| … After the Node struct is initialized, there are at least seven (I stopped counting after that) layers of N & P’s. Each N and P now has a Data, N and P pointers in it. It’s a deep, deep rabbit hole. |
|  |
|  |
| You can see the pointers in the 2nd node are pointing back to the head node |
|  |
|  |
|  |
|  |
|  |
| First run of list.getData(0) |
|  |
|  |
| Still in first run of getData() |
|  |
|  |
| Everything above looks fine until you expand that cascading head -> next chain where data now equals 4,2,1. And this keeps going on…. |
|  |
|  |
| Data should equal 1 as we are on the first iteration of the loop but clearly it shows 4.  It’s picking up the ‘4’ from the next -> data as seen in the previous pic |
|  |
|  |
| On the first iteration it’s calling the 4th draw method.  So something is going on here with the Node declaration. It looks like the pointers are pointing to other Nodes but that declaration with the Struct is not quite proper. |
|  |
|  |
| I changed the code in getData… from: return pointer-> next-> data;  to: return pointer-> data;  and you can see we are at least getting to the correct data value now in the first iteration of the loop. |
|  |
|  |
| So now we are getting the proper value in our loop |
|  |
|  |
|  |
|  |
|  |
| It looks like the pointers are sort of working in that they are pointing to different addresses, so that’s a good sign. |
|  |
|  |
| After pointer = pointer -> next the value of pointer does point to the next address from the previous pic:  0x00946be8, but this data value at that node is a 4 (from page 11). And hence that issue of the Node initialization is coming back to haunt as the value should be 2. |
|  |
|  |
| Second pass through loop gives a data value of 4, not correct. |
|  |
|  |
|  |
| Conclusion: There is something fishy about the Node struct and the thing to do might be to use a Node class like |