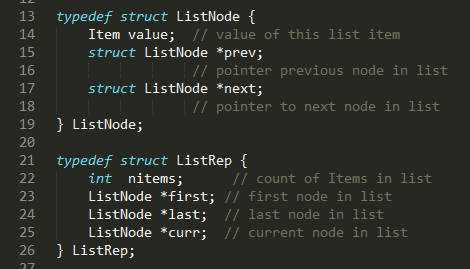
COMP1927 – Ass1

# What we did wrong

## Treat the header struct as if it used pointers

### The Problem

So, this is a huge difference in coding that we have to comply with (assignment spec), and we ignored. Consider the struct from lab03:

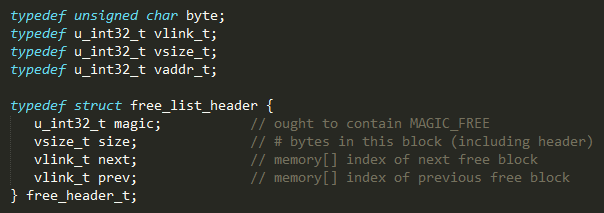


Note the pointers. Specifically, if you had a ListNode named Bob, and ran the following:

Printf(“%p”, Bob->next);

You’d get a memory address, something like 0x12345678 (hexadecimal, I don’t know that shit)

Now consider our struct:



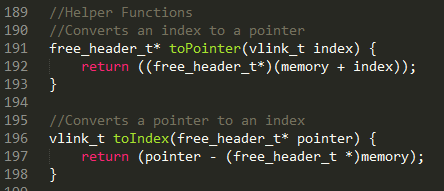
Note the terrifying lack of pointers (no \*s). By contrast, if you had a free\_header\_t named Bob, and ran the following:

Printf(“%d”, Bob->next);

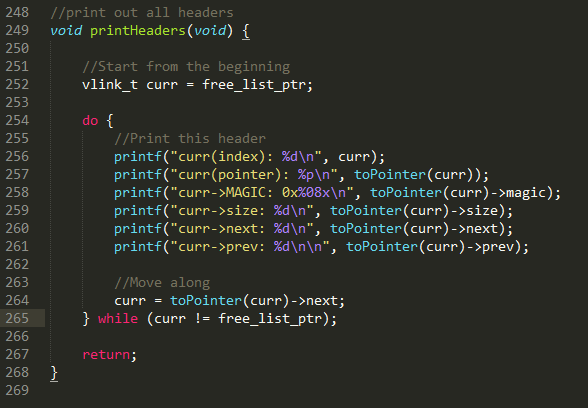
You’d get AN INTEGER, something like 4. THIS SUCKS DICK

### The Solution

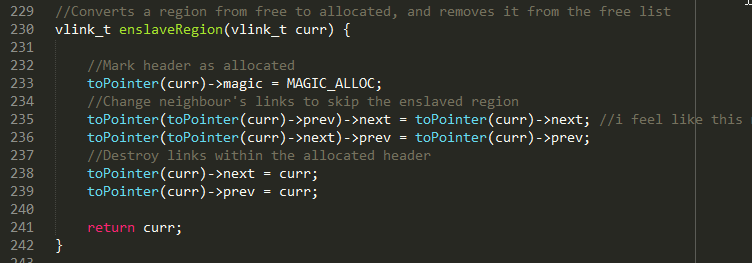
These babies:



These functions convert the index (which is useful as shit) into a pointer (which is useful), and vice versa. Here’s an example of them being used:



Basically, if you want to access the curr->next or curr->prev, any of the useful stuff – you need to wrap the curr in the toPointer function. Here’s a super gay example where the nesting makes this a headache:



\*Note that I’m not 100% sure this code is correct, although I’m pretty sure the use of toPointer is correct (it compiles at least)

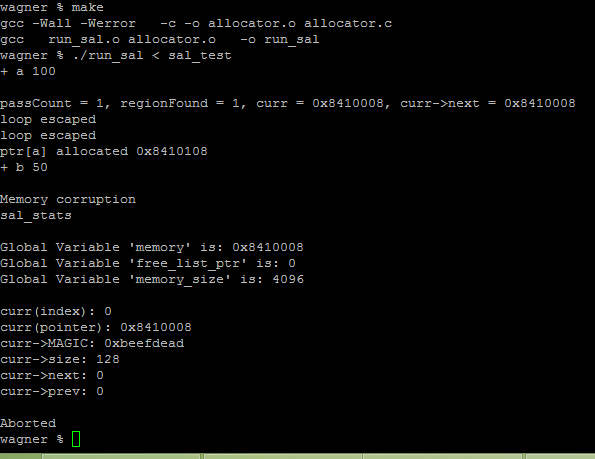
This is needed because curr is of type vlink\_t curr. Specifically – curr is an index (vlink\_t is a u\_int32\_t, which is an unsigned 32bit int). In previous exercises, we would’ve defined curr as a LinkNode or something to that effect, which handled numbers neatly. This time however, it’s just a sad old integer.

This took me a long while and much explanation to comprehend – it’s a fucking pain, but the program is compiling, and so far is at least allocating the first header (nothing beyond that).

## Not compile and test the program as we went along

Especially considering the size of this task, we needed to compile and test each section as we wrote it. It’s especially erroneous because they gave us a testing program, and debugging tools with sal\_stat. Before we start writing a new function, all of the existing ones should A. compile without error and B. actually work. The compile error printouts for allocatorBROKEN.c are insurmountable.

After writing any code, this is what we should have done:



That’s the latest test of the program, you can see that it aborts in sal\_malloc (look at the source), and generates a nice list of statistics to help solve the problem (in this case, the search for a free list came up with an allocated one). The file sal\_test.txt is in the ass1 directory, its currently just what was on the assignment page but we’ll have to upgrade it before we submit.

I think most of the code logic we wrote is correct, but importing it from allocatorBROKEN.c has to be done one step at a time, and each section tested and fixed.

## Not use github’s version control properly

So if you look at the last few hours – I’ve got 10 or so commits. The recommendation I received was that each “commit” operation within github should reflect some new feature – at least a new function() but often just a chunk of meaningful code.

Takeaway: commit far more often. It provides us with a better history to undo changes.

# What we need to do now

So, the furthest I got working on allocator.c is stored as allocatorBROKEN.c. This file is – broken, its not much more advanced from previous commits but it contains a lot of the toPointer stuff which is necessary to make allocator.c work. We need to migrate all the remaining code in broken to allocator, one function at a time, testing each one and making sure it works, fixing any errors we find.

## Functions

### Functions Fully Imported and Fixed

* Sal\_init
* Sal\_end
* sizeToN
* toPointer
* toIndex
* printHeaders

### Functions imported, but not yet working 100%

* Sal\_malloc
* EnslaveRegion
* memoryDivide

I’m actually not sure which of these work and don’t work, currently the program terminates within sal\_malloc but the problem is caused (I think) by one of the other two)

### Functions not yet imported

* Sal\_free
* Merge

I’ve got the shell of merge in there, so it doesn’t cause compile errors if you call it. Same for sal\_free.

## Obstacles

I’ve got an assignment due Wednesday 10am, and it’s mostly blank page, so I need to devote all my time to that until it’s done. I’ve put a message on OL clarifying the due date of this task (either Thursday or Friday midnight), but either way I have Thursday free, plus nights

Don’t worry too much about the lab – looks like a bitch but kinda fun. It’s low priority for now.

I’m working today 930am – 3pm, but as this is now my second night without sleep I’m not sure what state I’ll be in tonight, and I’ll probably still have that assignment to work on. Read the comments on the commits, they (poorly) explain what’s happened.