Please use spaces instead of tabs or convert the tab to spaces. Other people who view your code could have different settings for their tabs so it may look ugly on their side when they try to view it.

Heres how to do it on the two main editors :

Emacs ~/.emacs

(defun my-c-mode-hook ()
(progn
(setq c-basic-offset 2)
(make-local-variable 'compile-command)
(setq compile-command (concat "jikes -nowarn -g +E " buffer-file-name))
(message "JDE mode hook executed")))
(setq-default indent-tabs-mode nil)

VIM ~/.vimrc

set expandtab

set ts=2

set sw=2

Other editors :

IDK, sorry

Exercise today in class :

Introduced the typedef notion and how you can typedef the pointers to the struct. What this means is that you can make another name for struct\*. We introduced this to you in class because this appears multiple times in the Homework code. One instance that was brought up in section was the LinkedList. Remember that this name “LinkedList” is typedefed to a pointer, so don’t be thrown off by that.

* The main point of this exercise was to get you modularize the code by making separate files for the class definition of the object (.h for defining the methods, .c for implementation of the methods), and for the main function that uses the code.
* Remember that the #include is basically a copy and paste of the entire file into that one line. So you could leave out the #include “section2struct.h” in the .c implementation of the Section2Struct object code because when you #include “section2struct.h” in main it would have the function definitions in the code. In other words, this code would compile without warnings, but it is a really bad idea and is generally considered an error. During link time it would link the implementations to the existing code.
* A main use case for using a \*\* or another \* than necessary such as using int \* for an int, is if you want to mutate objects in other functions or if you want to have return parameters in functions. The homework does that by using \*\*payload. This allows for multiple return values in functions. This could be helpful in HashTable when trying to find an element in a chain.
* (More complicated!)Another use case is when you would want to allocate a multidimensional and want to use syntax like a[i][j] to access it subsequently. In that case, you allocate arrays of pointers to rows (for a 2d array), for instance, so the array itself is int\*\*.

\*payload = something..;

* C can have function pointers, you can invoke them the same way as regular functions. For example in you may see payload\_free\_fn pointers around the code. You can invoked it like payload\_free\_fn(…);

HOWEWORK

* Start early! It takes a while, mostly to get used to what exactly is going on