CT331 Assignment 1

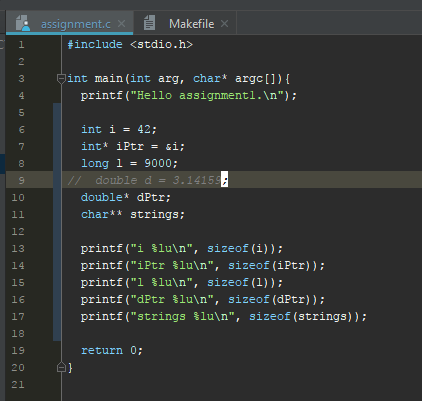
Andrew Reid East

16280042

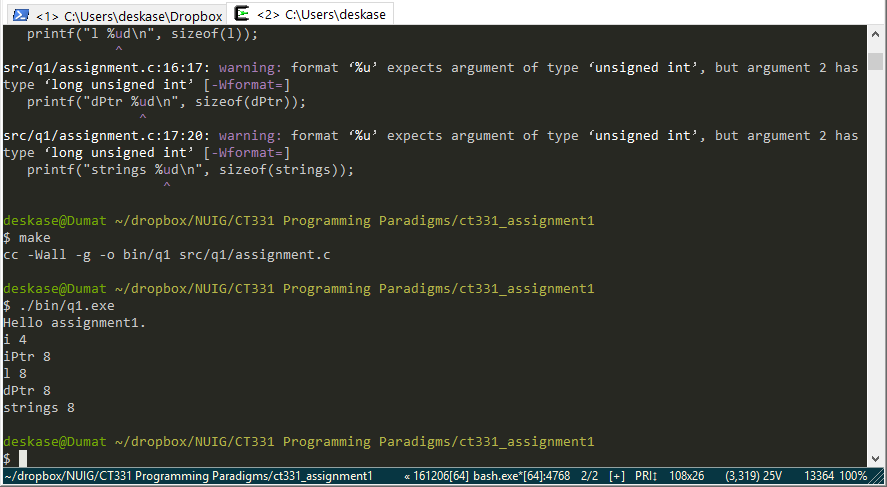
<https://github.com/reideast/ct331_assignment1>

# Question 1

## Code Editor



## Command Line Output



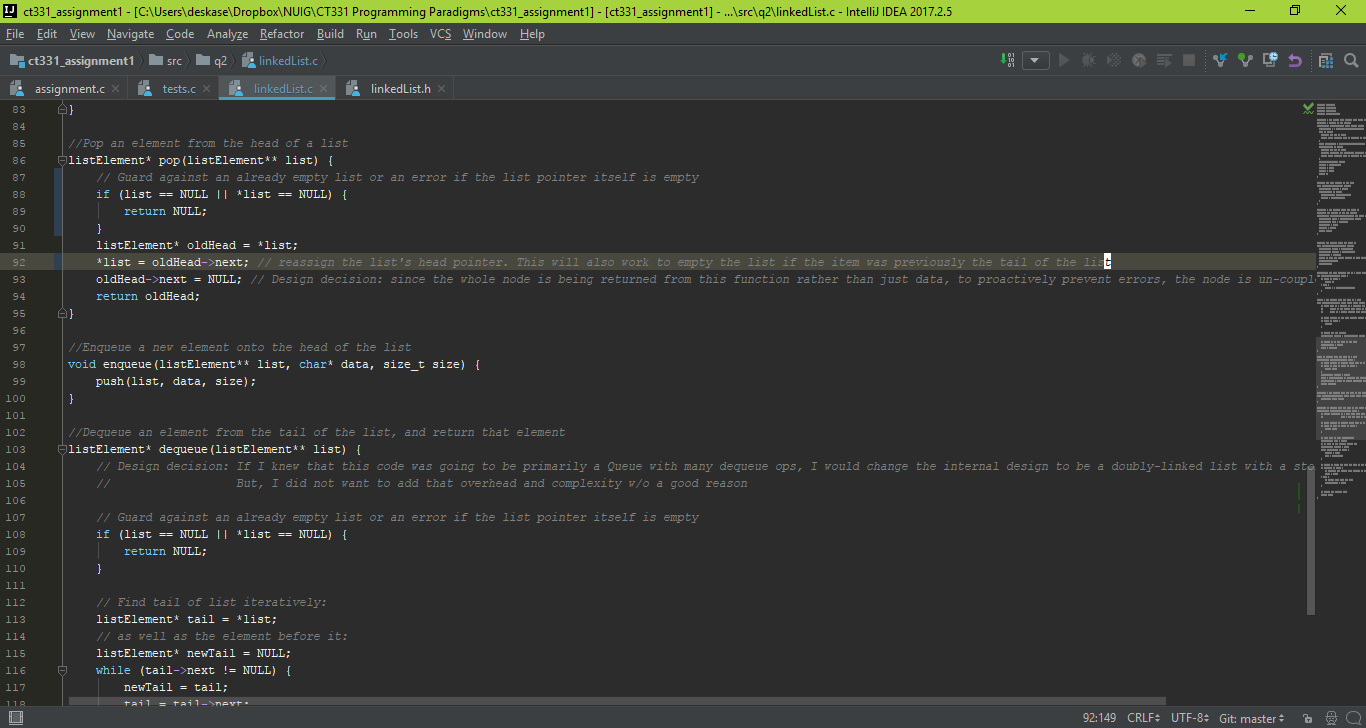
## Comments

The size of the stack data variables, the integer and the long are as expected: an int is 4 bytes, 32 bits, and a long is twice that much, 8 bytes. Four bytes is the most common integer size for modern compilers, and I expected to see a long be twice as much storage (although it is not required to be larger to be compliant).

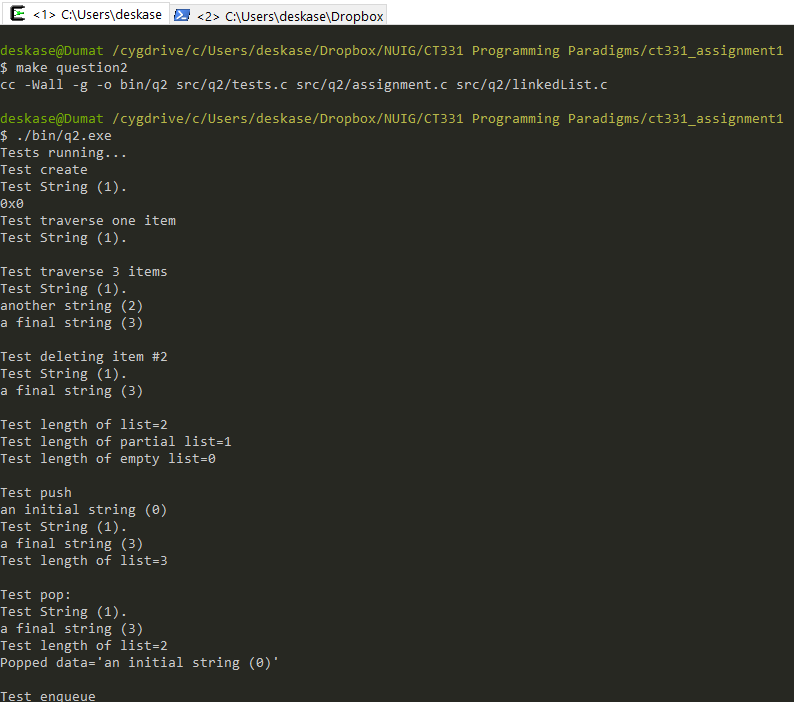
The amount of memory needed to store all of the different pointer types ended up being identical, even though the storage needed for the data they may point to is different. Since this was complied on 64-bit built Cygwin on a 64-bit Windows build on an x64 Intel processer, the 8 byte/64 bit pointer sizes was as expected. To further experiment, I installed a 32-bit build of Cygwin, (had to change the printf format specifiers for 32-bit gcc’s requirements) and then saw that in this case, 4 bytes were required for the pointers.

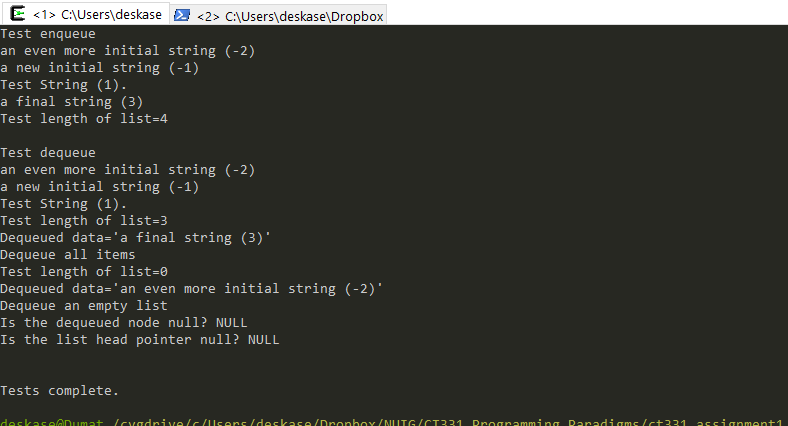
# Question 2

## Code Editor



## Command Line Output





## Comments

It was particularly interesting to figure out why push(), pop(), and enqueue() needed a double-star pointer to pass the head of the list as an argument, as I have not seen that idiom before. I briefly questioned to myself how push() could be implemented without returning a reference to the new head of the list, then I was confused when I saw the \*\*. Finally, after a complier error on dereferencing the pointer, I realized the connection, and knew the reasoning how to make the function work properly.

# Question 3

## Code Editor

## Command Line Output

## Comments