Andre Godinez

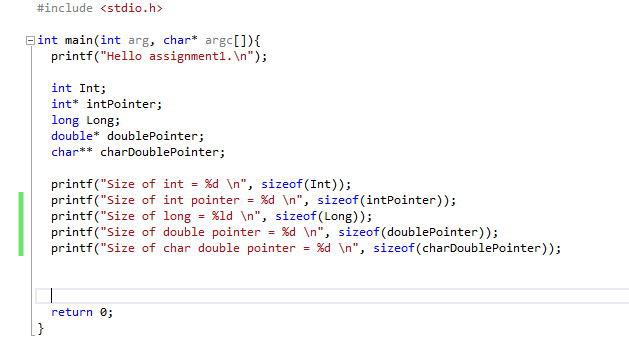
15460718

<https://github.com/Ajguy97/ct331_assignment1>

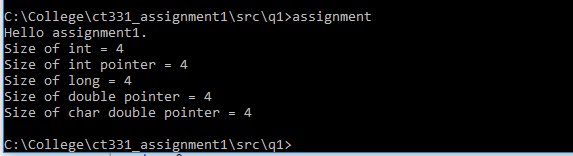
**Question 1:**

**(a)**

Screen shot of code:



Screen shot of cmd line output:



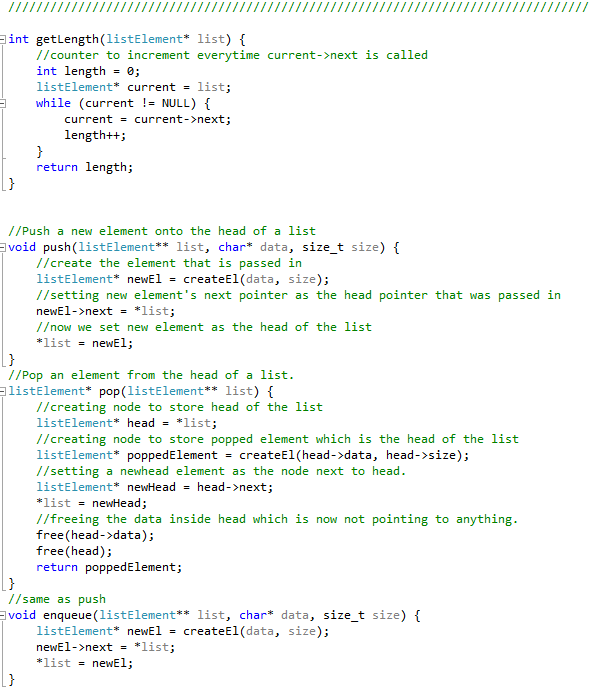
(b) Comment on results.

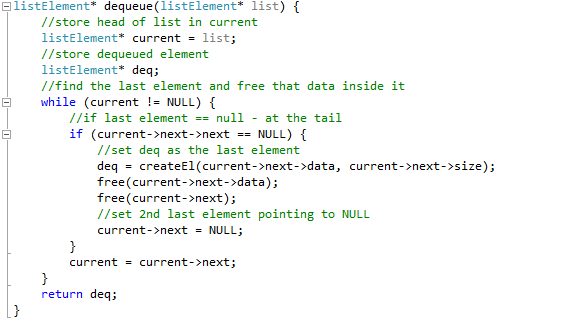
All data types shown have the same size. For int and long and other data types sizes are either 4 or 8 bytes depending on cpu architecture 32 bit on 64 bit and compiler settings. The same for pointers where the size is platform dependent on the computers processor architecture and the amount of bits you compile with. In this case my cpu is 64 bit but my visual studio is compiling in 32 bits therefore the sizes of all the pointers are 4. (in most cases 32 bits have size 4 bytes and 64 bits have 8 bytes.)

**Question 2:**

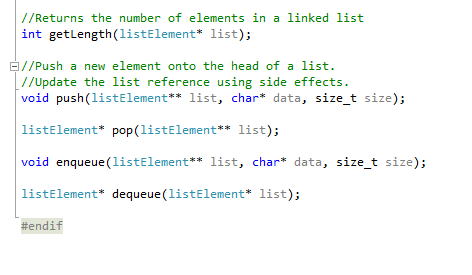
Screen shot of code :

**Q2: Linkedlist.c added code**

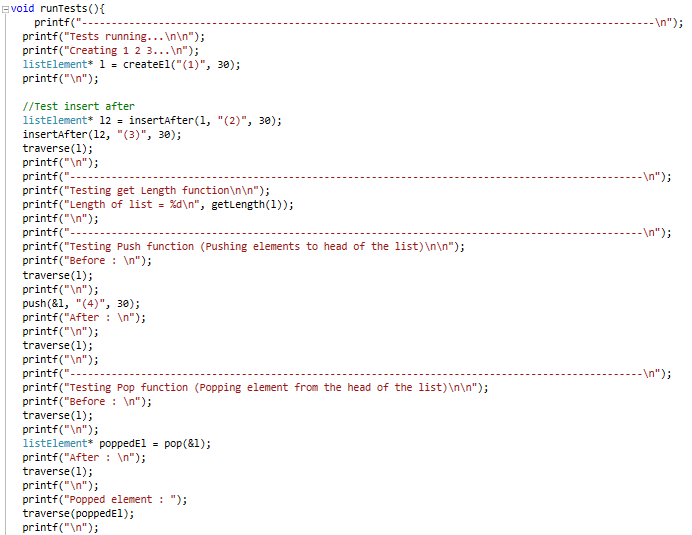


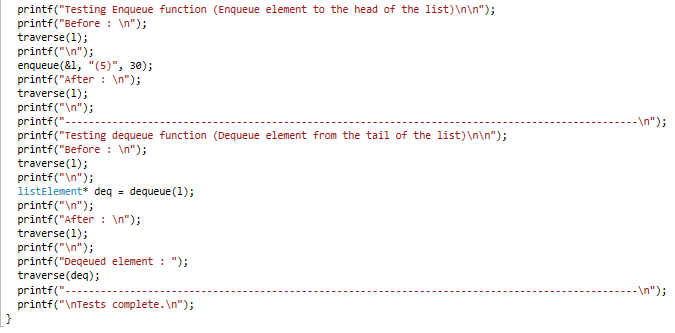


**Q2: Linkedlist.h added code**



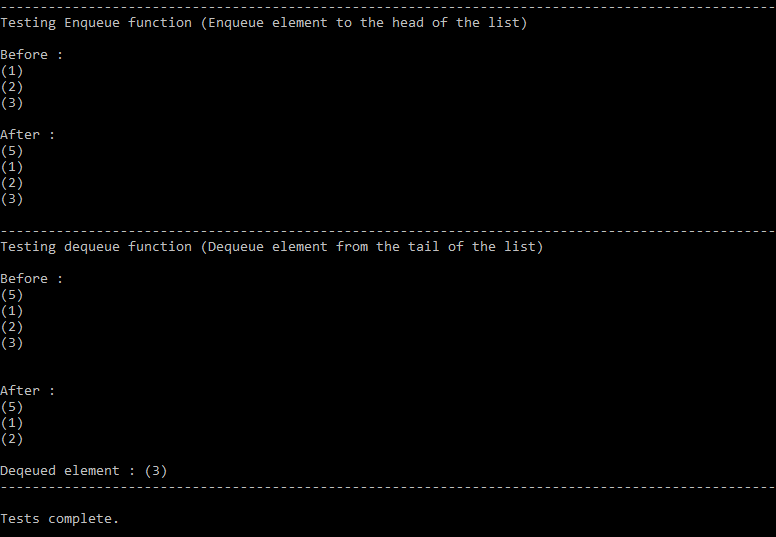
**Q2 : Tests.c added code**





**Q2 : Screen shot of cmd line output:**

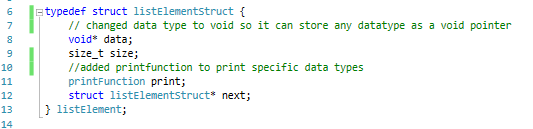


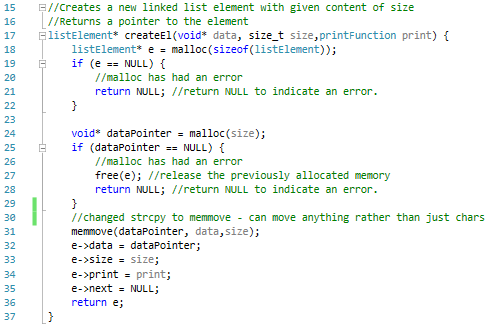


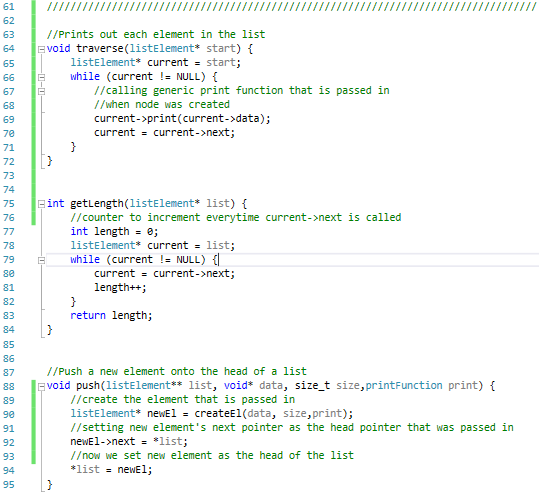
**Question 3:**

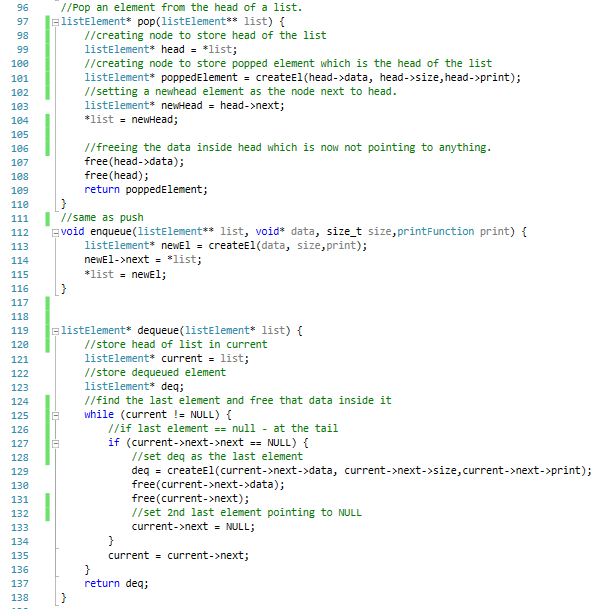
Screen shot of code :

**Q3 Genericlinkedlist.c added code:**

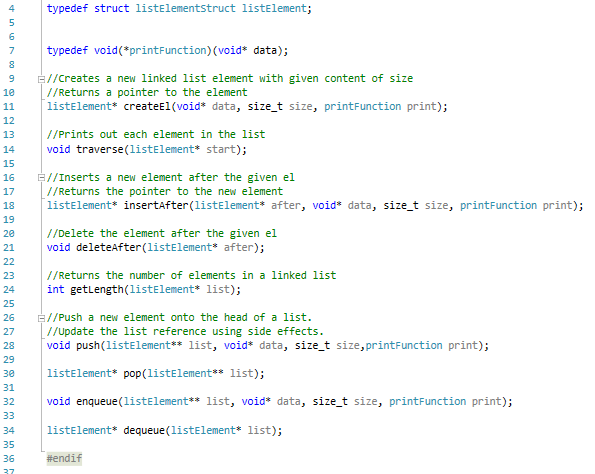




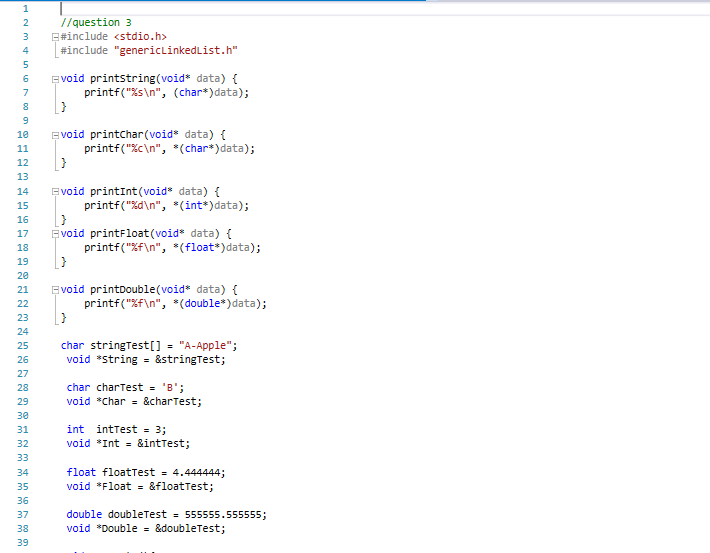


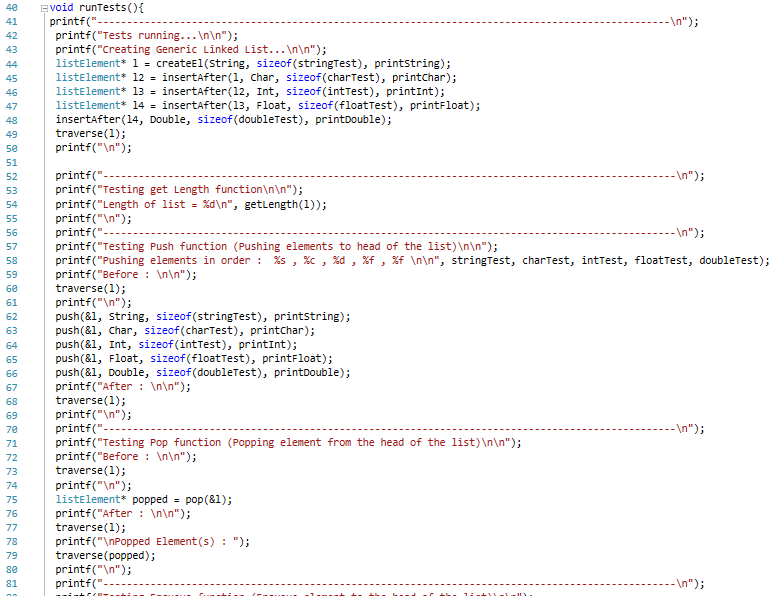


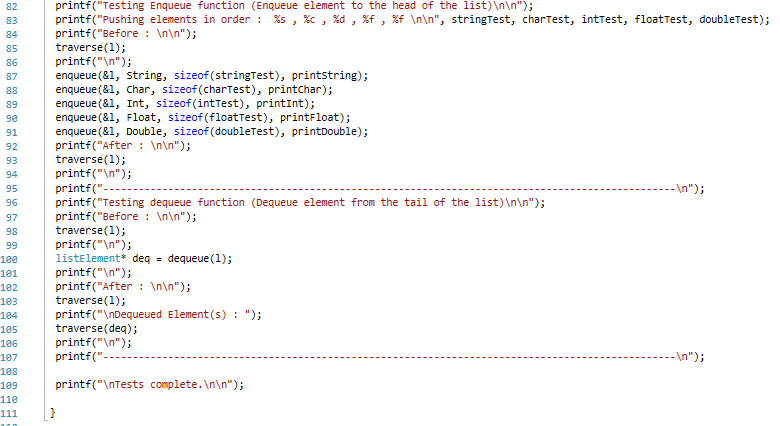
**Q3 Genericlinkedlist.h added code:**



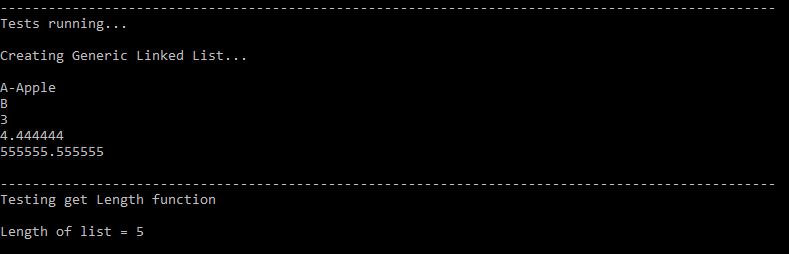
**Q3 Tests.c added code:**

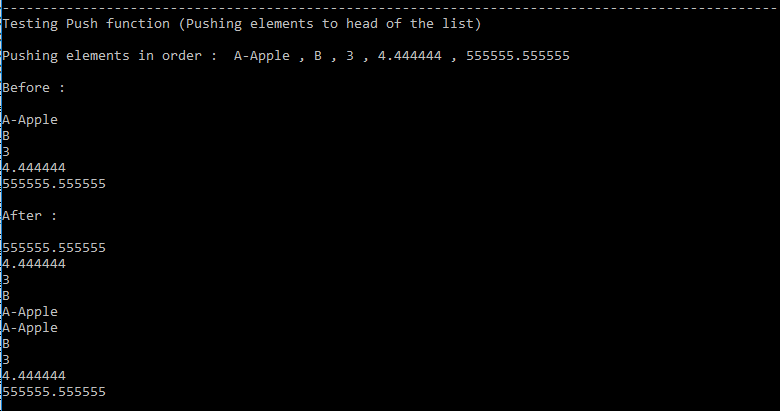


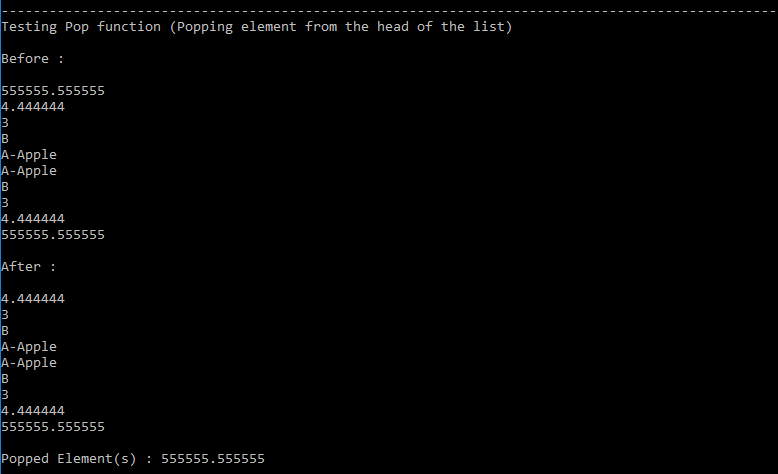


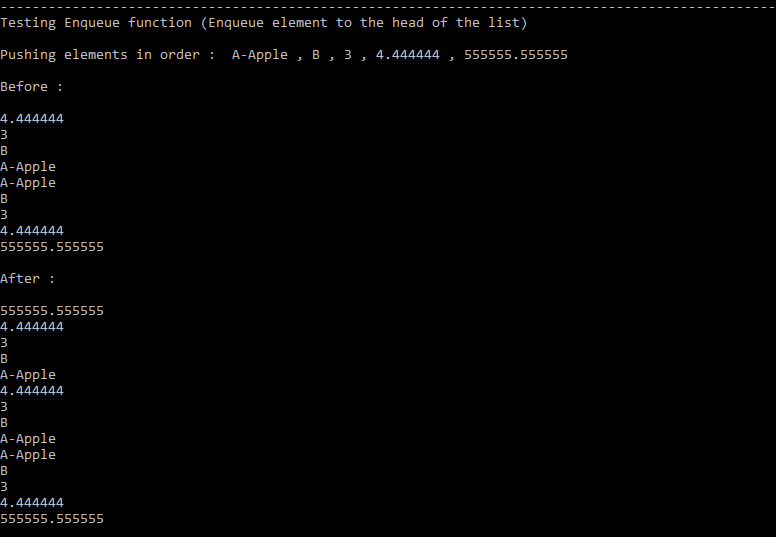


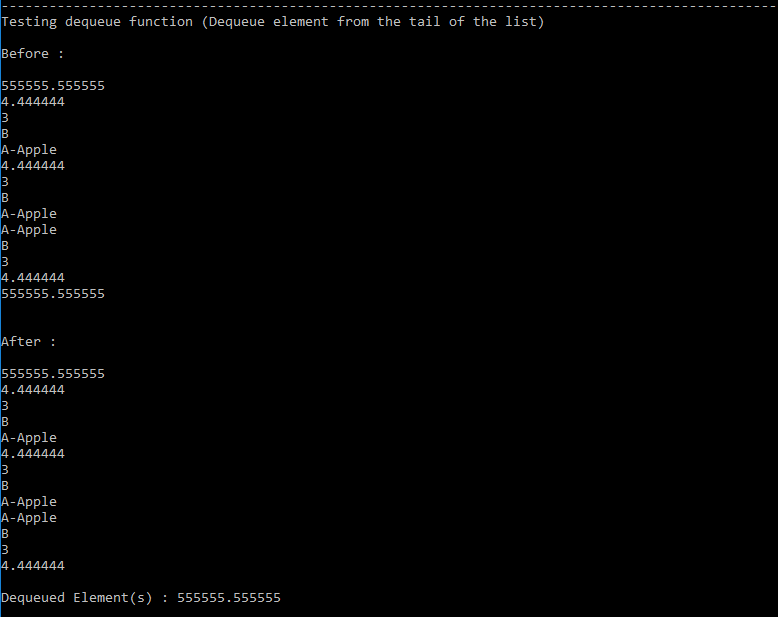
Screen shot of cmd line output:











**Question 4:**

**(a)**

Memory required to traverse a linked list in reverse tail to head

* Traversing from head to tail needs 1 node pointer and call node->next to get the next node in the list until next is NULL – Linear function therefore not as memory intensive.
* Traversing from tail to head
* Create a node – current, tail - set current as head pointer passed in.
* While (current->next !=null) – if it’s equal to null then only 1 node in linked list – return that node.

If(current->next == NULL || current->next == tail ) – that means this is the tail of the list

- Do whatever with current -> print it or return this node

- set tail as this node

- set current as head pointer passed in again.

* Current = current->next
* This function is recursive and depends on amount of nodes in the linked list therefore more memory required.

**(b)**

How could the structure of a linked list be changed to make this less memory intensive?

* Usage of doubly linked list.
* Node has a pointer to next & previous node.
* Allows traversal head to tail & tail to head without recursion in functions. Eg. Current->next & current->prev