**Linked List Assignment**

Write a program that builds an ordered linked list of ***numbers greater than zero***, until the user inputs 0 or a negative number – then your program should exit. **If you sort the data and then build the list from the sorted data, you will lose at least 50% of your credit. If you rearrange the data in the nodes instead of ordering the nodes, you will lose at least 50% of your credit. Extraneous code will reduce credit.**

1. Use a struct that looks like:  
   struct listNode {  
   int data; **//ordered field** struct listNode \*next;};

I am not kidding, use the Header node - not a pointer to the first node!!!! If you missed the classes where I discussed Header nodes, then you need to get the notes from one of your classmates.

1. Use a dummy listNode named Header to point to the beginning of your list:

**struct listNode Header = {0, NULL};**   
  
The dummy node means that you will not need to have code for **SPECIAL CASES**   
(for insertion into an empty list, beginning of list, end of list, etc.).   
If you put in code for special cases, you will lose points.

1. Here is a list of all the possible struct pointers that you may need:   
     
   newNode (for use with malloc( ))  
   current to keep track of where you are in the list  
   previous to keep track of the node before current, since you need to do insertions  
   temp to use with the free( ) to deallocate the nodes you created using malloc  
    **Header is not dynamically allocate, so don’t try to free( ) it!**

Do not use typedef for any reason!!

1. Use the malloc( ) function in <stdlib.h> to dynamically allocate new structures  
    **#include <stdlib.h>**
2. Include a function that prints out the list using pointers. **Run this automatically before the program EXITS. Also, be sure and free( ) your nodes!** Do the freeing in a separate function that runs just after printing and before exiting.

**Hand in:**

* + Code printout, with your name, class, etc.
  + Follow the “rules” for assignment 1 – e.g.
    1. user interface with your name and description of the program
    2. labeled output
    3. all the other stuff
* Source code in Black Board drop box.

**Make your pointers global, so that you don't have to worry about passing them to functions - this will make things much easier for you - I promise!**

**Your user interface MUST work like this:**

Have the user input positive numbers until they are finished – input zero or negative to end.

Do Not ask them for anything else, do not give them any other options – this I am going to be running scripts to test the programs – if yours does something else, if will fail in the test phase.

Functions:

1. main( )
   * Should either generate the user welcome and instructions or call a function that does.
   * Calls the function(s) to create the new node and insert it into the list.
   * Calls the function to print the list just before freeing and exiting.
   * Calls the function to free the list, then exits.
2. create new node ( )
   * Create a new node and initialize it.
   * Can also insert the node, or you can do this in a separate function.
3. insert node ( )
   * This could be done in the last function.
4. print the list ( )
5. free the list ( )