WESTERN UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Software Tools and Systems Programming
(Computer Science 2211a)

LAB 8
The week of November 09, 2020

The purpose of this lab is to demonstrate the use of pointers to create a doubly linked list. Although we know from class that 'pointers' do not actually 'point' to anything, we know that by using memory addresses we can create user define types of variables and connect heterogeneous structures into a chain construct.

This lab will create a doubly linked list and print the list out displaying the value stored and the number representing the position in the list when it was entered.

PREPARATION:

Review the class slides on linked list.

LAB 08:

Create a C library

The goal is to create a create a doubly linked list populated with randomly generated integer values and print the list out along with the position number of each link (element) of the list

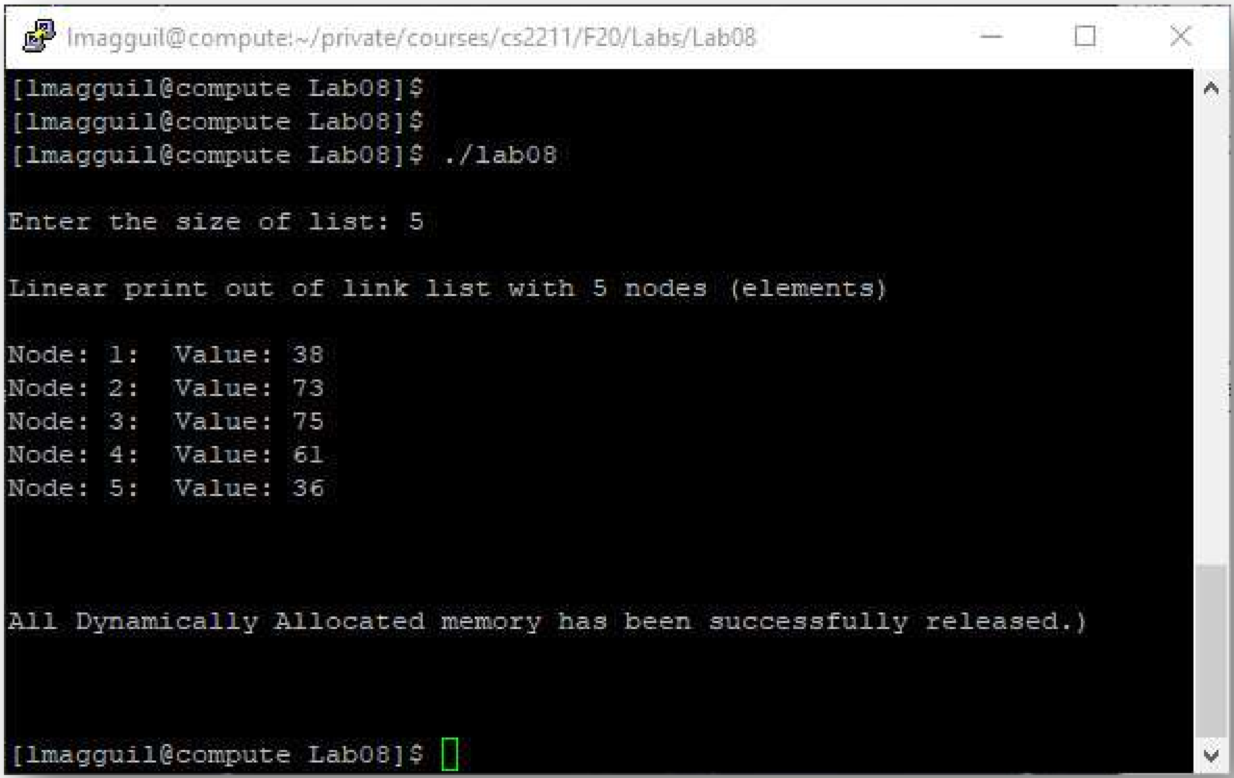
The concepts explored are variable reference, void pointers and pointers.

INSTRUCTIONS:

This program will create a **doubly linked list** (a list consisting of elements that 'point' (reference) the element (aka link or node) **before** (previous) and the element (aka link or node) **after** (next).

Each node in the double linked list will point to a structure that contains two integer variables. One of these variables will hold a randomly generated integer value and the other will represent the position of when the node was created. So the first will contain 1, the next will contain 2, then 3, etc. representing the position of that data element when it was inserted in the doubly linked list.

The program will then print this list out in the order it currently exists. This output will print each node. This output will show the node number and the contents of the integer variable.



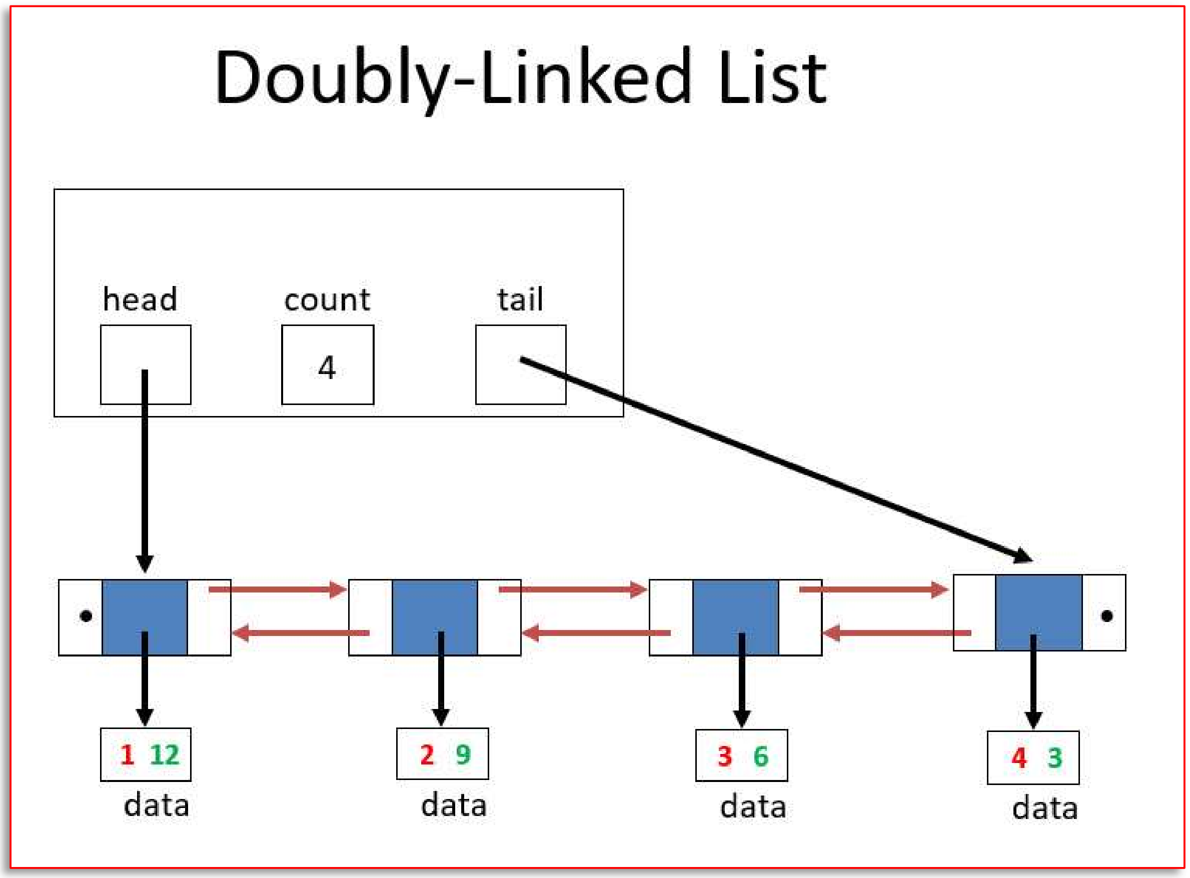
```
[lmagguil@compute Lab08]$
[lmagguil@compute Lab08]$
[lmagguil@compute Lab08]$ ./lab08

Enter the size of list: 5

Linear print out of link list with 5 nodes (elements)

Node: 1:  Value: 38
Node: 2:  Value: 73
Node: 3:  Value: 75
Node: 4:  Value: 61
Node: 5:  Value: 36



All Dynamically Allocated memory has been successfully released.)



[lmagguil@compute Lab08]$
```

Review and use the code in the notes as a guideline to creating your program.

The program will free the dynamic memory allocated for the list.
This includes the memory allocated for the data structure AND the memory allocated for the link.

The structure and definitions for this program are expected to follow the examples in the notes to create a generic double linked list through the use of the void pointer in each node to point to data consisting of any variable type (primary and/or user defined).

Each node will have a void pointer to a data element. This data element is a **structure** consisting of two integers. One holds the element (link) position in the list (i.e. 1 for 1st element in the list, 2 for second element, etc.) indicating the position when the node was added to the list and the other holds the randomly generated number assigned to this element.



Complete the function to create the list (notice the user is prompted for the size of the list):

Once created a separate function will be called to print out this doubly linked list:

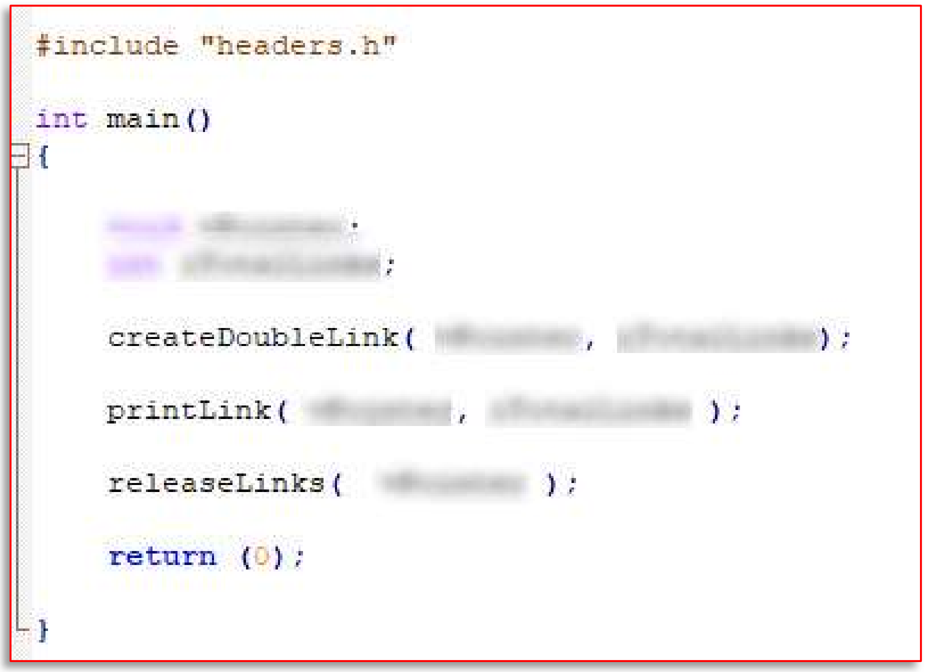
Each line will contain the position number (shown in red above) and the integer value (shown in green above) stored in that node.
It will require indirect addressing and type casting to achieve this.

You are to use the C function of `rand()` to compute the random number.

Finally, you will be required to complete the function to release the any and all dynamic memory you allocated for this program

The program is to be modular in design and the main program should be a control program only.

The suggested (you do NOT have to use this exactly) code for the main is:



```c
#include "headers.h"

int main()
{

    [redacted] [redacted];
    [redacted] [redacted];

    createDoubleLink( [redacted], [redacted]);

    printLink( [redacted], [redacted] );

    releaseLinks(  [redacted] );

    return (0);

}
```

The files in my solution were:

  headers.h
  definitons.h
  Utilities.c
  DLLInputOutput.c
  main.c

hint: these file names are only (merely suggestions). You do NOT have to use any or all of these names (except of rmain.c of course).
You can break down the code in any manner you see fit, as long as they are not all in a single main.c

Compile and run the code:

Required Coding Standards

All code is to be indented correctly.
Comments at the very beginning (top – first lines) of each file must be:

```
/* CS2211a 2020 */
/* Lab 08 */
/* your name */
/* your student number */
/* your UWO Account Name */
/* Date Completed */
```

FINISH:

Submit (upload) your finished work the <Assignments> section of OWL under Lab 08.

Submission Instructions:

Change your current working directory to ~/courses/cs2211a/Labs. Create file YourUserName_lab7.tar.gz and submit this file for lab 7. (For detailed information, please check *CS2211a Lab Submission Guidelines*).