CS 1037a – Computer Fundamentals II
Assignment 5

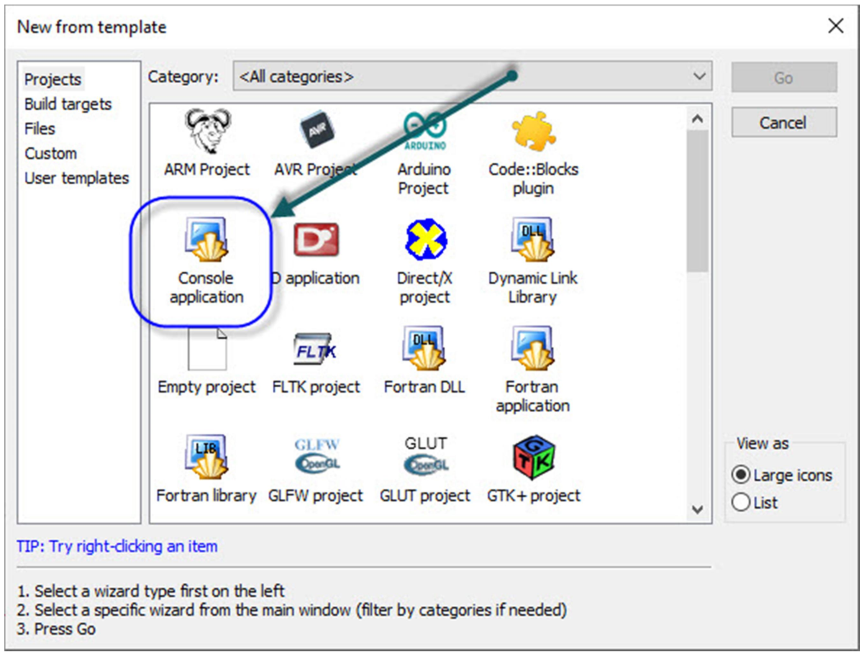
Instructor: L. Magguilli
Due Date: Tuesday, December 03, 2019, 6:00 pm **(yes, extended from the course outline)**.
Total marks: 100
Percentage of final mark: 5%
Accepted up to ONE (1) day late.

The purpose of this assignment is to put together all the pieces of the class that we have covered.

PREPARATION:
Create a new project in Code::Blocks.

File->New->Project.

1.) Select the <Console Application> option from the [New from Template] screen.
2.) Select the C (NOT the C++) language
3.) Name the Project Title: *YourAccountName*_Asn05 (using your actual user name).
4.) Use the default compiler by clicking on the <Finish> button.



1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

Under [Sources] in the Project window of Code::Blocks, open the main.c file.

The assignment will again be contained within multiple files.
To complete this assignment, take the code you completed in Lab10 and two functions.

Add code that will sort the doubly linked list using the Bubble Sort algorithm.

BUT! This time – you will change the nodes into the correct order (not just swapping the integer values).

SO! if the data item that node 4 points to is greater than the data item that node 3 points to, then change (swap) the nodes [NOT JUST THE NUMBERS].

ie.

NODE:	1	2	**3**	**4**	5
VALUE:	8	10	14	12	21

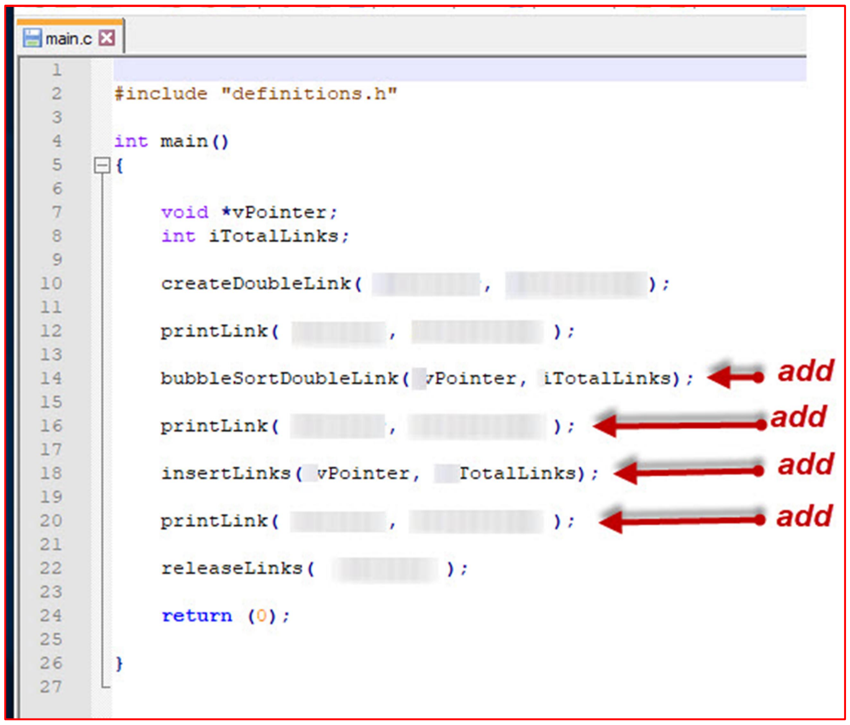
NODE 3 and NODE 4 have to be swapped.
SO --- swap the NODES. (NODE 4 and 3 switch places)

This will result in the list:

NODE:	1	2	**4**	**3**	5
VALUE:	8	10	12	14	21

The nodes are to be swapped.
This means changing the ->next and ->previous values as discussed in class.

The main.c from Lab10 will now look like:



```c

#include "definitions.h"

int main()
{

    void *vPointer;
    int iTotalLinks;

    createDoubleLink( [redacted] , [redacted] );

    printLink( [redacted] , [redacted] );

    bubbleSortDoubleLink( vPointer, iTotalLinks);

    printLink( [redacted] , [redacted] );

    insertLinks( vPointer, TotalLinks);

    printLink( [redacted] , [redacted] );

    releaseLinks( [redacted] );

    return (0);

}

```

The new functions are:

bubbleSortDoubleLink()
insertLink()

```
/* Function to sort a linked list using bubble sort*/
void bubbleSortDoubleLink( varType Var1, varType var2 )
{

  . . . . .
}
```

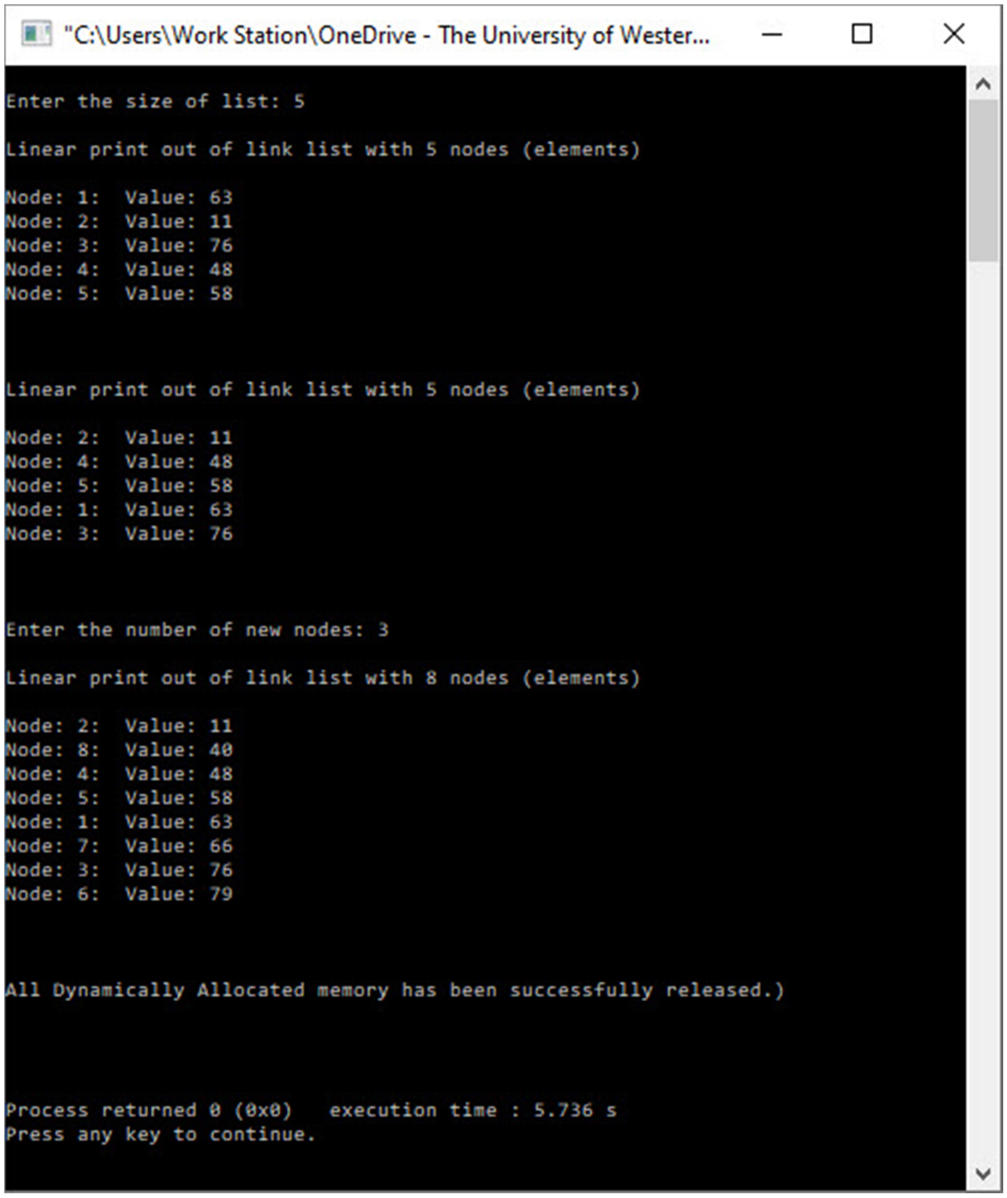
and

```
/* Function to insert n nodes to the existing list*/
void insertLinks( varType Var1, varType var2 )
{

    printf("\nEnter the number of new nodes: ");
     scanf("%d", &aSize);

    for ( int j = 0; j <aSize; j++)
   .....
        insertNode(var1, var2);
   .....


}

/* Function to insert a node into a linked list in the proper
position*/
void insertNode( varType Var1, varType var2 )
{
......
}
```

insertLink() will prompt the user for how many nodes to insert into the existing list.
It will then call **insertNode()** where the program will insert the new node into the existing list at the correct location (maintaining the list in ascending order.

Print out the list in between each of these steps (see the main.c code).
To show the total number of nodes (elements, links) in the list, the program uses a local variable named **iTotalLinks**. (hint: this is the second variable in the functions… passed by reference).

The output of this program looks like:



"C:\Users\Work Station\OneDrive - The University of Wester...

```
Enter the size of list: 5

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

Node: 1:  Value: 63
Node: 2:  Value: 11
Node: 3:  Value: 76
Node: 4:  Value: 48
Node: 5:  Value: 58



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

Node: 2:  Value: 11
Node: 4:  Value: 48
Node: 5:  Value: 58
Node: 1:  Value: 63
Node: 3:  Value: 76



Enter the number of new nodes: 3

Linear print out of link list with 8 nodes (elements)

Node: 2:  Value: 11
Node: 8:  Value: 40
Node: 4:  Value: 48
Node: 5:  Value: 58
Node: 1:  Value: 63
Node: 7:  Value: 66
Node: 3:  Value: 76
Node: 6:  Value: 79



All Dynamically Allocated memory has been successfully released.)




Process returned 0 (0x0)   execution time : 5.736 s
Press any key to continue.
```

(Notice, these are random number, so your numbers WILL be different.

Required Coding Standards

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

```
/* CS1037a 2019 */
/* Assignment 05 */
/* your name */
/* your student number */
/* your UWO Account Name */
/* Date Completed */
```

All variables MUST have a comment describing their intended use(s).

A comment describing the code for each function must be included.
The comment(s) can be brief, but must convey what that section of code performs.

ALSO – again a very important standard!
You are **NOT** to use the code

```
break
```

-or-

```
continue
```

anywhere in your code (except if you have a `switch` statement in your code).

This is bad coding practice and leads to 'spaghetti code' (with apologies to my Italian heritage…) where program execution control can go any which where.
We do not want to get into the habit of using these constructs. All the code can be written in a fluid manner without terminating the processes with a break or continue.

Also, any parameter in italics should be changed to a meaningful name. Do NOT use parameter1, parameter2 or paramter3 in your actual program.

Submission Instructions:

You must upload and submit, via the CS1037A OWL Web Site, all the files required for your program to compile and execute.:

yourUWOAccountName_Asn05.cbp
main.c
{ all other header files, etc. }

(example: assume my UWO email is kdoit373@uwo.ca
i.e. if my email is – **kdoit373@uwo.ca** then my ID will be – **kdoit373**
So, my UWO Account Name is: **kdoit373** and this assignment is **Asn04**.
therefore, my file names that are to be used for submission are:
kdoit373_Asn05.cbp -and- **main.c**

It is the student's responsibility to ensure the work was submitted and posted in OWL.
OWL replies with a summation verification email (every time).

Any assignment **not** submitted correctly will **not** be graded.

NOTE: ONLY the required files are to be submitted.
Do **NOT** submit any of the other files or folders.
Marks will be deducted if you submit anything more than just the requried files.
You **MUST** submit a .cbp Project File that can be opened in Code::Blocks
- if you could not get Code::Blocks to work on your computer, use the computers in the lab or in Middlesex College.

The teaching assistant grading your assignment will compile and run your program.
If the program does not compile, the TA will NOT attempt to correct or fix your program so it will run.

AND AGAIN PLEASE: Do not cheat or copy.
Remember, these are NOT community projects but are expected to be completed individually.
You will very definitely know how to do this for the final….