# Operating System: Winter 2019
# CSC 360
# Assignment 01 - Part 1
# Second Rev

January 15, 2019

## 1 Doubly linked list

A doubly linked list is a data structure that consists of set of nodes. Each node contains three variables: two pointers as links (referencing the previous and the next node) and one variable containing any type of data. Previous and next links of the beginning and ending nodes, respectively, point to NULL in order to represent termination of the list.

### 1.1 Simplified Interface

A simple implementation of the doubly linked list is provided in the following gist: https://gist.github.com/mexuaz/a0014cf8836900ab36e86ce6fe8dd45a

All the methods in this implementation have at least one reference to a arbitrary node of link list (not preferably head or tail of list) to apply the relevant function.

## 2 Assignment

There are also some other methods that are not implemented but the function signature is provided. Fairly brief and comprehensive comments are given for all methods including those not implemented. Using comments you have to complete two functions *swap* and *shift_left*. Notice that you are not allowed to change the implemented functions. And the functions you are trying to implement should be efficient.

1. void swap(struct node* ref1, struct node* ref2);
2. struct node* shift_left(struct node* ref, int n);

The swap function exchange the position of two nodes in the d-linked list. Important! DO NOT EXCHANGE THE VALUES FOR IMPLEMENTING SWAP. This is an efficient version of swap and since the underlying data could be big (not integer like this example) exchanging values instead of nodes might be very costly. The argument order tells nothing about nodes order and consider that they could be next to each other or in distant (You might need different solutions for each scenario).

Most of other standard algorithms such as different types of sort, reverse and ... use swap function in their body. The implementation of inverse is provided

to you. employing the inverse function you could test the correctness of your swap method.

The shift left function simply shifts the nodes one position to the left.

Those students seeking extra points might try following methods as well.

1. void reverse(struct node* ref);
2. struct node* concat(struct node* ref1, struct node* ref2);
3. struct node* unique(struct node* ref);
4. int distance(struct node* ref1, struct node* ref2);
5. struct node* rotate_left(struct node* ref, int n);
6. void minmax(struct node* ref, int* min, int* max);
7. bool includes(struct node* ref1, struct node* ref2);

## 2.1 Running and testing your code

In order to compile the source code use this command 'ssh username@linux.csc.uvic.ca' in Linux bash to login to 'linux.csc.uvic.ca' server. If you are using Windows or Mac OS you need to download Putty software instead. After you're logged in, use *nano* or *emacs* to create an empty C file called 'doublylinkedlist.c' and copy-paste the provided sample code. You could also simply run wget https://gist.githubusercontent.com/mexuaz/a0014cf8836900ab36e86ce6fe8dd45a/raw/a0558e930112d71f321fb81546954efc4e386c67/doublylinkedlist.c and download the entire file to your working directory in Linux.

Another option will be to use https://coliru.stacked-crooked.com (preferred) or http://cpp.sh/ as an online compiler. Use command 'gcc -std=c11 -pedantic-errors doublylinkedlist.c -o doublylinkedlist' to compile your source to binary. Using switches '-std=c11 -pedantic-errors' for compiling are mandatory in order to make your source code conform the ISO/IEC 9899:2011 standard. Your code should compile without any errors and warnings. After compiling you may run your code with command './doublylinkedlist' to see if it could provide the expected outputs. Also a sample usage for the d-linked list provided in the main file with the expected outcome in the comments at the beginning of the source code make sure you get the right answer.