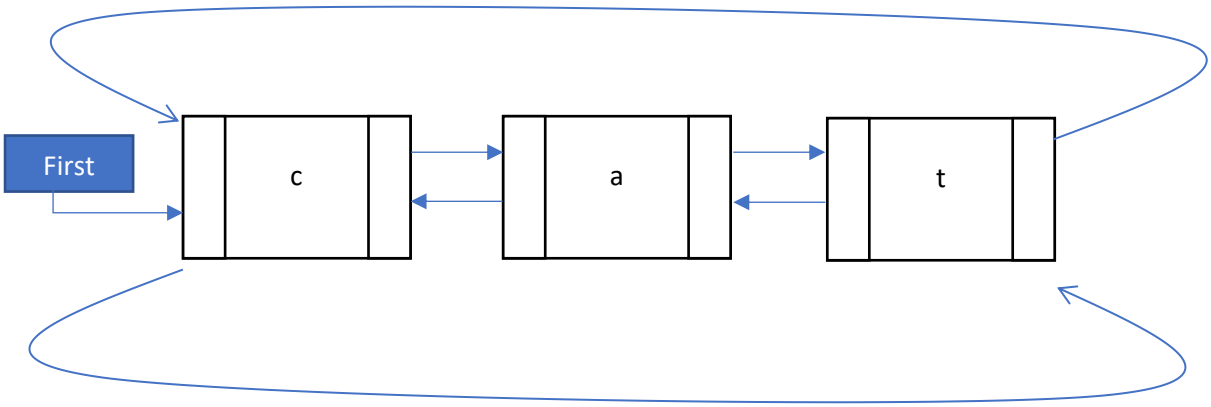
# Preliminary Task Provisions

- **THE ANSWER ARE TYPED NEATLY** and **ATTACH THE SCREENSHOT OF THE CODE AND THE OUTPUT FOR ALGORITHM QUESTION**
- Print the answer and paste it on B5 notebook. Write your identity on the cover of the notebook.
- PT is MANDATORY, IF YOU DON'T DO THE PT THEN YOU ARE NOT ALLOWED TO ATTEND PRACTICUM
- **SUBMIT WITHOUT ANSWER = NOT ALLOWED TO ATTEND PRACTICUM.**
- **PT BOOK MUST BE BROUGHT AS A REQUIREMENT TO ATTEND PRACTICUM**
- Deadline : Monday, 24 February 2020, 08.03 WIFLAB
- **THERE IS NO TOLERANCE FOR LATE SUBMIT**
- **PLAGIARISM IS NOT ALLOWED (PLAGIARISM = E)**
- Work with the PT clearly to make it understandable
- For every algorithm questions, you must include **NAME** and **SID** as follows.
- For every algorithm questions, insert your **SID** to filename **(Example : header_130416XXXX.h)**
- **FILENAME UPLOAD ONLINE : MODX_NIM_CLASS.rar/zip**

```
int example (int a, int b) {

  /*

  Name : Ichi Ocha
  NIM : 1301123456

  */
```

# PRELIMINARY TASK MODULE 5
## DATA STRUCTURE COURSE 2019/2020-2

Consider the following list:




**Part I:** Make circular double linked list ADT above in "cdll.h"

Data type definition for the following list:

```
type infotype: char

type address: pointer to elmtList

type elmtList: <info: infotype, prev: address, next: address>

type list: <first: address>
```

The following subprogram definition:

```
function isEmpty (L: list) → boolean
{the function returns true if the list is empty, and false if it isn't}

procedure createList (output L: list)
{IS. –
 FS. defined L, list is empty}

procedure createNewElmt (input X: infotype, output P: address)
{IS. X is info that will be placed on the new element to be allocated
 FS. Defined list element with address P, where info from P is X, or NULL if the new element
allocation is fails}

procedure insertFirst (input/output L: list, input P: address)
{IS List L may be empty. P has been defined to be inserted into L,
F.S. P becomes the first element of list L.}

procedure insertAfter (input Prec, P: address)
{I.S. Prec is not NULL and is an element list L. P will be inserted after Prec
F.S. P has been inserted into L and located after Prec}
```

procedure *deleteFirst (input/output L: list, output P: address)*
*{I.S. L has one or more elements*
*F.S. The first element has been removed and recorded on P. List L may be empty}*

procedure deleteAfter (input Prec: address, output P: address)
*{I.S. Prec is an element list L, Prec^.next doesn't refer to the first element, and Prec^.next may refer to the last element*
*F.S. The element after Prec has been removed and recorded on P}*

function countWord (data: array of character, L: List) → integer
*{The function returns the number of words found, and returns 0 if not found. Hint: Please googling for writing arrays in parameters}*

procedure printInfo (input L: List)
*{IS: List may be empty*
*FS: Display all info in list L if the list is not empty. If the list is empty, display the message "List is Empty"}*

**Part 2:** Make a circular double linked list implementation above in "cdll.cpp".

**Part 3:** Make a main program in "main.cpp" to test the functions and procedures that you have created, with the following insertion and deletion scenarios:

Display all data *{Expected output: List is empty}*

Insert 'a' as the first element

Insert 't' as the first element

Insert 't' after 'a'

Insert 'c' after 't'

Insert 'a' as the last element

Insert 's' as the last element

Insert 'c' as the last element

Insert 'a' as the last element

Displat all data *{Expected output: t c a t a s c a}*

Find word "cat" *{Expected output: 2}*