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, 17 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

  */
```

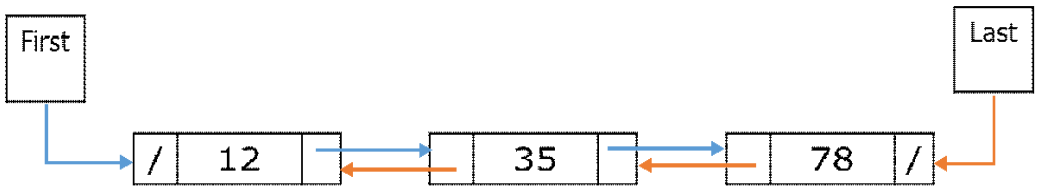
PRELIMINARY TASK MODULE 4

DATA STRUCTURE COURSE 2019/2020-2

Double Linked-List with First and Last pointer

PRELIMINARY TASK

Consider the following list:



PART I: Make an double linked list ADT above in "doublelinkedlist.h".
Definition of the data type for the list:

type infotype: integer
type address: pointer to elmtList
type elmtList: <info: infotype, prev: address, next: address>
type list: <first, last: address>

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

procedure createList (output L: list)
{I.S. –
F.S. defined L, list is empty}

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

procedure insertFirst (input/output L: list, input P: address)
{I.S List L may be empty. P has been defined to be inserted into L; P ^ .next and p ^ .prev = NULL
F.S. P becomes the first element of the L list}

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

procedure insertLast (input/output L: list, input P: address)
{I.S List L may be empty. P has been defined to be inserted into L; P ^ .next and p ^ .prev = NULL
F.S. P becomes the last element of the L list }

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

procedure deleteAfter (input/output L: list, Prec: address, output P: address)
{I.S. Prec is a list L element and Prec^.next is not NULL. Prec^.next may point to the last element
F.S. Element after Prec has been removed from L and recorded on P}

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

procedure concat (input L1, L2: list, output L3: list)
{I.S. L1 and L2 each have one or more elements
F.S. L3 contains all L1 elements combined with all L2 elements}

function median (L: list) → real
{I.S. L has ascending ordered elements
F.S Looks for the median in L list. L list is not empty. The function returns the median of the L list. If the number of elements in the list is odd, the median is the info value of the list element in the middle position, whereas if it is even, then the mean info value of the two list elements in the middle}

PART 2: Create a double linked list implementation above in "doublelinkedlist.cpp".

PART 3: call all functions and procedures in the main program "main.cpp" to prove the functions and procedures give correct results.