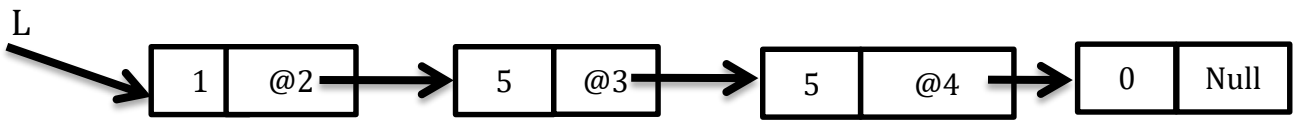
Lab Session No. 01 – Linked Lists

Recall

A Singly Linked List (SLL) is a set of nodes, dynamically allocated and linked together. A node is a record composed of two fields: the element field containing the information, and the next field containing the address of the next node. A linked list is represented by a pointer containing the address of the first node of the list. An empty list, with no elements, has the reference NULL.



Definition of the List Type in C++

```
struct maillon {
   type_of_elements ele;   // declaration of a node in C++
   maillon *next;
};
typedef maillon *list;      // definition of type 'list'
```

Example: Declaration of an integer linked list

```
#include <iostream>

struct maillon {
   int ele;
   maillon *next;
};

typedef maillon *list;

int main() {
   list L = NULL, L1;
   ...
}
```

Primitive Operations on Linked Lists

void add (list &L, int x); // adds an element at the head of list L
list add (list L, int x); // adds as a function
int first (list L); // returns the first element of list L
bool is_empty (list L); // checks whether L is empty
list rest (list L); // returns the address of the second element
int length (list L); // returns the number of elements in list L

Exercise 1

(Program template available on the e-learning site)

We consider a singly linked list of integers declared as shown below. You are asked to:

1. Complete the program by defining the following functions:
   - add: adds an element to the list L
   - display: displays all the elements of the list L
   - sum: returns the sum of the elements in list L

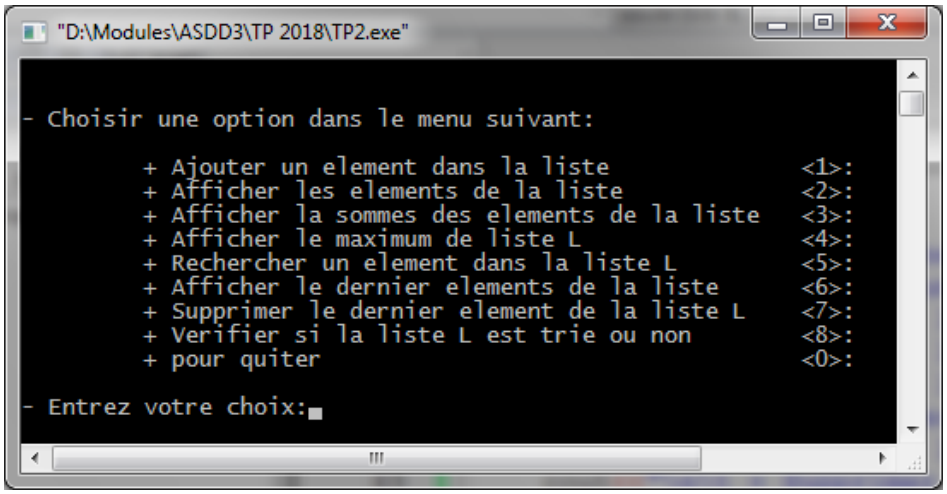
2. Using iterative functions (loops), improve the program to also:
   - Display the maximum element of list L
   - Search for an element in the list
   - Display the last element of the list
   - Delete the last element of the list
   - Check whether the list L is sorted or not.

```
/**********************    TP2.cpp      **************************/
# include <iostream>
using namespace std;
/***************Définition du type liste************************/
struct maillon{
    int ele;
    maillon *suivant;
};
typedef maillon *liste;
/***************déclaration fonctions*****************************/
void afficher_menu();
void ajouter (liste &L, int x);      // ajoute un element dans la liste L
int premier (liste L);               // retourne le premier élement de la liste L
void afficher (liste L);             // affiche tous les élements de la liste L
int somme (liste L);                 // reourne la somme des éléments de la liste L

/**************** Le programme principale ***********************/
int main()
{
liste L=NULL;   int choix;
do
{   afficher_menu();
   cin>>choix;
   switch (choix)
   {
```

```
        case 1: int x; cout<<"Entrez un entier X:  "; cin>>x; ajouter(L, x);  break;
        case 2: cout<<"les elements de la liste sont: "<<endl; afficher(L); break;
        case 3: cout<<"la somme des elements de la liste est :"<< somme(L); break;
    }
}while (choix !=0);
  system("pause") ;
}
/****************définition des fonctions************************/
void afficher_menu()
{
    system("cls");
    cout<<"\n\n- Choisir une option dans le menu suivant:";
    cout<<"\n\t + Ajouter un element dans la liste                  <1>:";
    cout<<"\n\t + Afficher les elements de la liste                 <2>:";
    cout<<"\n\t + Afficher la sommes des elements de la liste      <3>:";
    cout<<"\n\t + pour quiter                                      <0>:";
    cout<<"\n\n- Entrez votre choix:";
}
```

Example of Final Program Output



Exercise 2

Modify the program from Exercise 1 by defining recursive versions (instead of iterative ones) for the functions previously implemented using loops.