Lab 3: Structures
Thursday October 2nd /Tuesday October 7th
SITE - University of Ottawa Fall 2025
Due ONLINE Wednesday, October 22 at midnight
In groups of up two students and only One submission is required per group
/10

I. Introduction
This Lab assignment deals with enumerations, structures, linked and Self-referential structures.

II.1. Structures
Example :

```
struct count
{
	int number;
	char state;
	char name[80];
	float balance;
};
```

Declaration of variables of *count* structure type :

```
count clientl, client2;
count client[100]; /* client array of  100 count type items*/
```

II.2. Passing a structure to a function:
Example using pointers :

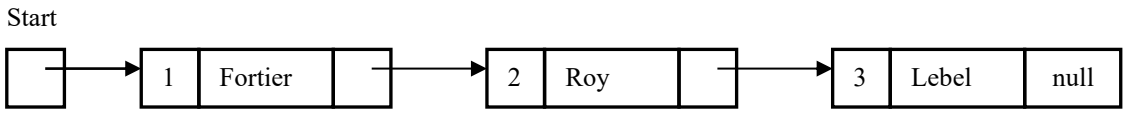
```
#include <iostream>
#include <string.h>

using namespace std;

struct Enrollement {
	char name[10];
	int number;
};

void maj(Enrollement* pe);	//maj function

int main() {
     Enrollement client = { "Durand",666 };//client variable of  Enrollement type
     cout << client.name << endl;
     cout <<client.number << endl << endl;
     maj(&client);		//calls maj function
     cout << client.name << endl << client.number << endl;
}

void maj(Enrollement* pe){
	strcpy_s(pe->name, "Dupont");
	pe->number = 999;
}

/*Output*/
Durand
666

Dupont
999
```

II.3. Self-referential structures

Self-referential structures are very useful in applications implementing chained data structures such as lists. We will only deal with simple linked lists in this Lab. The basic idea is that each item in the list contains a pointer to the next item. The interest consists in a simplified management of the list; the withdrawal, the addition of an element is done using pointers. The definition of a self-referential structure in a simple list using a structure is done as follows:

```
struct list
{
	int number;
	char name[80];
	list* next;
};
```

The "**next**" member is a pointer to a structure of the same type. By creating several variables responding to this kind of structure, we can organize the following linked list:



III. Lab Assignment 3

Exercise 1: (4 Marks: 2 for each question)

a) Complete the attached C ++ code (*main* in **myFile1a.cpp**) to build a deck of 32 cards. The game is represented by an array of 32 distinct elements, each of the Card type. You must submit your file **myFile1a.cpp** completed.

b) Write the missing function called *testPair()* (attached **myFile1b.cpp**) to determine if a poker player has a pair. The function receives a hand (an array of five different cards) as input and displays true if and only if all five cards have at least one pair (a pair is formed by two cards of the same face (ex: two jacks)).
You must hand-in one file **myFile1b.cpp**.

```
/*Output*/
I have at least: 1 pair
```

Exercise 2: (6 Marks)

The exercise consists of writing a program that allows you to initialize a simple linked list (linked data structures) with the following data:

student
student grade

After entering the first item in the list, the program offers the following menu (See attached file):
1) Display of the complete linked list.
2) Add a new item to the list.
3) Remove an item from the list.
4) Calculate the grades average of all students.
5) Exit the program.

Please find attached the *main* program and the corresponding header file (myLinkedList.cpp and myLinkedList.h).

Questions : **(6 Marks)**

Write the following missing functions to the program:

1) *add* function: (1.5 Marks)
This function adds a new item to the list. The new element is inserted in the list in a position defined by the user, by entering on the keyboard the number of the element which precedes it. The validity of this number must be verified by the program.
We create the element inserted in the list with memory reservation for this element. Its member data (student and grade) are entered using the keyboard.
The *add* function takes as arguments, a pointer to a structure of type *Evaluation*, and the number of existing elements in the string.
It returns a pointer to a structure of type *Evaluation* which corresponds to the first element of the linked list.

2) *remove* function: (1.5 Marks)
This function removes an item from the list. the number of the element to be deleted is entered on the keyboard and must be valid.
The *remove()* function takes as arguments, a pointer to a structure of *Evaluation* type and the number of existing elements.
It returns a pointer to a structure of *Evaluation* type which corresponds to the first element of the linked list.

3) *display* function: (1.5 Marks)
This function displays the complete linked list, student and grade of each item (One item per line).
It takes as arguments, a pointer to a structure of *Evaluation* type.

4) *average* function: (1.5 Marks)
It calculates the average mark for all students and displays it.
It takes as arguments, a pointer to a structure of type *Evaluation* and the number of existing elements.
It returns 1 if all goes well, 0 otherwise.

Complete these four functions in the same file **myLinkedList.cpp and submit it**.

```
/*Example of output*/
```

1) Display of the complete linked list.
2) Insert an element
3) Remove an element.
4) Calculation of the class average.
5) Exit the program.

Your choice ?:2
After which element you want to insert ? (0 for start): 0

Entering the item from the chained list.

Enter the student: Jane Benoit
Enter the grade: 80

1) Display of the complete linked list.
2) Insert an element
3) Remove an element.
4) Calculation of the class average.
5) Exit the program.

Your choice ?:2
After which element you want to insert ? (0 for start): 1

Entering the item from the chained list.

Enter the student: Jack Fortier
Enter the grade: 75

1) Display of the complete linked list.
2) Insert an element
3) Remove an element.
4) Calculation of the class average.
5) Exit the program.

Your choice ?:1

Student :Jane Benoit
The grade is :80

Student :Jack Fortier
The grade is :70

1) Display of the complete linked list.
2) Insert an element
3) Remove an element.
4) Calculation of the class average.
5) Exit the program.

Your choice ?:4

The average of the class is: 75

1) Display of the complete linked list.
2) Insert an element
3) Remove an element.
4) Calculation of the class average.
5) Exit the program.

Your choice ?:3
what is the number of the element to delete ?: 2

1) Display of the complete linked list.
2) Insert an element
3) Remove an element.
4) Calculation of the class average.
5) Exit the program.

Your choice ?:1

Student :Jane Benoit
The grade is :80

1) Display of the complete linked list.
2) Insert an element
3) Remove an element.
4) Calculation of the class average.
5) Exit the program.

Your choice ?:5

Submit your work ONLINE (one zip file only) before Wednesday, October 22 at midnight

Instructions

- Create a directory that you will name Assignment3_ID, where you will replace ID with your student number (the one submitting the assignment).
Put all the following files in your compressed directory Assignment3_ID.zip for submission in the Brightspace Virtual Campus.

Files :

- ✓ *README.txt*
- ✓ *myFile1a.cpp*
- ✓ *myFile1b.cpp*
- ✓ *myLinkedList.h*
- ✓ *myLinkedList.cpp*

- Don't forget to add comments in each program to explain the purpose of the program, the functionality of each method and the type of its parameters as well as the result.
- In the Assignment3_ID directory, create a text file named README.txt, which should contain **the names of the two students**, as well as a brief description of the content:

Student Name:
Student Number:
Course Code: CSI2372A

Academic Fraud:

This section of the assignment aims to raise students' awareness of the problem of academic fraud (plagiarism). Consult the following links and read both documents carefully:
https://www.uottawa.ca/current-students/academic-regulations-explained/academic-integrity
University regulations will apply to all cases of plagiarism. By submitting this assignment:
1. You confirm that you have read the above documents;
2. You understand the consequences of academic fraud.