UNIVERSITÉ DE MONTPELLIER

FACULTÉ DES SCIENCES DE MONTPELLIER

MASTER DE MATHÉMATIQUES

PARCOURS *"Modélisation et Analyse Numérique"*

PROGRAMMATION II - HAX011X - 2025/2026

CC2 - SPARSE POLYNOMIALS

The mandatory deliveries are specified in red color. Provided and helper files are specified in blue color. Please, respect the naming convention. The use of language model-based chatbots is strictly forbidden.

Let consider the polynomial $r(x) = x^{100} + 5x^2$.

Storing it in the list-based polynomial class of CC1 would lead to a quasi-empty list of size 101, containing mostly zero coefficients (only two terms would be non-zero). Thus, it would be more suitable to store only the non-zero coefficients, in a more flexible data-type based on the linked_list class.

In a more general setting, let consider the polynomial $p(x) = \sum_{k=0}^{d} a_k x^k$. For each non-zero monomial term $a_k x^k$ in this sum, two informations should be stored: k and a_k. In order to store only the informations associated with the non-zero monomials, let introduce a simple class called monomial, as follows:

- the coefficient $a_k \in \mathbb{R}^*$ is stored in a T member called value_,
- the power $k \in \mathbb{N}$ is stored in a **size_t** member called power_.

Hence, this monomial class is (partly) defined as follows:

```
template<class T>
class monomial
{ protected:
      size_t power_;
      T value_;

  public:
      monomial():power_(0.){};
      monomial(size_t nn, T aa):power_(nn), value_(aa){};
      size_t power() const{return power_;};
      T value() const{return value_;};
  ...
};
```

and is provided in a file called class_monomial.hpp. We also provide a class class_linked_list.hpp and a main.cpp file. Your goal is to design a class sparse_polynomial, in a file class_sparse_polynomial.hpp.

Exercice 1. A class sparse_polynomial

This class should **derive** from the class linked_list<monomial<**double**>> and have the following members and features:

(1) two constructors should be provided, with the following prototypes:

```
sparse_polynomial(); // default cstr
sparse_polynomial(monomial<double> const& monom , linked_list<monomial<double>>* pt = nullptr)
```

(2) a destructor ~sparse_polynomial(),

(3) a method with the following prototype **size_t** degree() **const** should returns the degree of the stored sparse_polynomial. As a convention, **the higher-order term of the polynomial is stored as the first** coefficient,

(4) your class should compile and link with the provided main.cpp program,

(5) you can print the polynomials as linked_lists but it is not pretty. Define a new version of **operator**<<() for your new class sparse_polynomial, so that the command std::cout << r << std::endl; results in printing r in line as follows:

```
x^100 + 5 x^2
```

Exercice 2. Sum of sparse_polynomials

Most of the methods of the linked_list class can be straightforwardly applied to the sparse_polynomials. However, one has to define the sum of two sparse_polynomials:

Define, in the class_sparse_polynomial.hpp file, an overloaded **operator**+() which allows to sum two sparse_polynomials and return the sum as another sparse_polynomial. In particular, the commands

```
sparse_polynomial p4(monomial(4, 9.));
sparse_polynomial p6(monomial(1, 10.));
 rr = p4 + p6;
```

should make sense and return the expected sparse_polynomial. To achieve this, you should use the **operator**+=() of the linked_list class, and modify it accordingly to perform the summing operation whenever needed. The resulting file should be called class_linked_list_modified.hpp,

Your resulting files should compile and link with the provided main_sum.cpp. We recall that according to the assumptions made in the merging algorithm, the two sparse_polynomials should be correctly ordered, and so should be the resulting one.

Exercice 3. Evaluation of sparse_polynomials

Define, in the class_sparse_polynomial.hpp file, an overloaded **operator**()(**double const**& x) which allows to evaluate a sparse_polynomial at the given point x and return the value. In particular, the command

```
std::cout << "p6(2)=␣" << p6(2.);
```

should return the correct value. Your resulting file should compile and link with the provided main_eval.cpp.

As a summary, two files should be delivered:
class_sparse_polynomial.hpp, class_linked_list_modified.hpp.