PW 6: Linked List (Individual Work)

Goal

- Define an abstract data type
- Handling dynamic memory
- Write and test subprograms (always!)
- Write explanations (always!)

Instructions

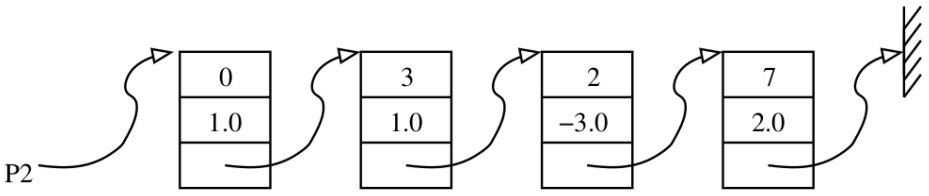
The objective of these exercises is to program some operations that manipulate polynomials. We are interested here in polynomials with a real variable, here are some examples:

$$P_1(x) = 2x^2 + x + 1$$
$$P_2(x) = 2x^7 + x^3 - 3x^2 + 1$$
$$P_3(x) = x^{415} - 3,1x^{55}$$

As the polynomial P3 illustrates, we want to be able to represent polynomials of very high degree but having few non-zero coefficients. As a result, we decide to store only the monomials with a non-zero coefficient by chaining them together. Next image gives a possible representation of the polynomial P2 in memory.

No order is imposed on the monomials. However, there cannot be two different monomials of the same degree.

Even if we decide not to store monomials with zero coefficient, it is possible, in particular following an operation, that some monomials with zero coefficient appear.



Note that the various subroutines requested can be written in iterative and recursive mode.

It will be necessary to estimate the complexity (in the worst case) of all the written subroutines.

Exercise 1: Define type (typedef)

Define the types necessary to represent the polynomials respecting the choices above. Specify in particular the invariants on these types.

Exercise 2: Initialize

Write a subroutine to initialize a polynomial. The polynomial is then zero (P (x) = 0).

Exercise 3: Destroy

The destroy operation will be used when the programmer knows that he will no longer have to use a polynomial.

3.1 Explain why it is necessary to define (and think about using!) This operation.

3.2 Can we use the initialize operation to zero a polynomial? Justify the answer.

3.3 Is it necessary to use destroy on a local variable of a subroutine? Justify the answer.

3.4 Write the “destroy” subroutine.

Exercise 4: Display

Write a display_debug subroutine that displays monomials in the order they appear in the representation of a polynomial. For example, applied to the polynomial represented in previous image, the subroutine will display:

```
-->(1.0 X 0)-->(1.0 X 3)-->(-3.0 X 2)-->(2.0 X 7)--E
```

The parentheses allow to highlight each monomial. The objective of this operation is to display the in-memory representation of the polynomial and thus better control and verify its evolution during the execution of the program. It will therefore be useful for adding traces in programs.

Exercise 5: Nullity of a polynomial

Write a subroutine that indicates whether a polynomial is the zero polynomial or not.

Exercise 6: Add a monomial

Write a subroutine to add a new monomial to a polynomial. This sub-program therefore performs the operation:

$$P(x) \leftarrow P(x) + cx^n$$

Even if this operation results in canceling the coefficient of a monomial, we will keep the monomial in the list (with a zero coefficient, of course).

Exercise 7: Add two polynomials

Write a subroutine that gives the sum of two polynomials. He therefore performs the operation:

$$R(x) \leftarrow P(x) + Q(x)$$

Note that neither P (x) nor Q (x) are modified by this operation.

Exercise 8: Remove null monomials

Operations on polynomials can introduce monomials with zero coefficient. This new operation aims to clean up the representation of a polynomial by removing all its monomials with zero coefficients.

Exercise 9: Putting a polynomial in canonical form

Write a subroutine that puts a polynomial in canonical form. That is, its monomials are ordered in descending order of degrees. Of course, monomials with zero coefficients are removed.