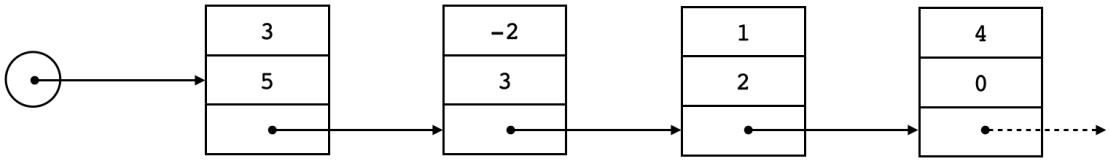
Assignment 2: Ada Programming (25%)

POLYNOMIAL ARITHMETIC

A polynomial is a type of algebraic expression in which the exponents of all variables should be a whole number. The standard form of a polynomial refers to writing a polynomial in the descending power of the variable. For example:

$x^2 + 2x + 5$

It is possible to store polynomials as a linked list. Here is what a linked list to store a polynomial would look like:



$3x^5-2x^3+x^2+4$

- Each node represents one term of a polynomial and is a structure containing a coefficient, an exponent, and a pointer to the next term of the polynomial.
- The terms of every polynomial are stored in the order of decreasing exponent within the linked queue, and no two terms have the same exponent.
- Terms with zero coefficient are not stored in the polynomial.
- The polynomial that is identically 0 is represented by an empty list.

Polynomials behave a lot like integers. Just as we can add, subtract, or multiply two integers and the result is always an integer, we can add, subtract, or multiply two polynomials and the result is always expressible as a polynomial.

- The basic rules for adding numbers or polynomials is the same. The only difference is when dealing with the addition of polynomials, like terms needed to be paired up, and then added together. Otherwise, all the rules of addition from numbers translate over to polynomials. Here is an example:

$$\begin{aligned} &(x^3 + 7x^2 - 5) \\ +\ &(4x^3 - 2x^2 + 6x + 9) \end{aligned}$$

$$
\begin{array}{l}
\hline
= (x^3 + 4x^3) + (7x^2 - 2x^2) + 6x + (-5 + 9) \\
= 5x^3 + 5x^2 + 6x + 4
\end{array}
$$

- Subtracting two polynomials is very similar to subtracting two numbers. One easy way to subtract polynomials is, just change the signs of all the terms of the polynomial to be subtracted and then add the resultant terms to the other polynomial as shown below. We just have to align the given polynomials based on the like terms.

$$
\begin{array}{l}
\quad (3x^2 - 5x - 1) \\
- (2x^2 + 3x + 2) \\
\qquad - \quad - \quad \text{(signs altered)} \\
\hline
= (3x^2 - 2x^2) + (-5x - 3x) + (-1 - 2) \\
= x^2 - 8x - 3
\end{array}
$$

- To multiply to polynomials, we just multiply every term of one polynomial with every term of the other polynomial and then add all the results. Here is an example to multiply polynomials.

$$
\begin{array}{l}
(5x + 1) \times (2x^2 + 3x) \\
= 5x(2x^2 + 3x) + 1(2x^2 + 3x) \\
= 10x^3 + 15x^2 + 2x^2 + 3x \\
= 10x^3 + 17x^2 + 3x
\end{array}
$$

TASK

Write a program in Ada to perform arithmetic operations on two polynomials.

1. Create an Ada **package**, `polylink` to deal with the input, output and storage of the polynomials using linked lists. The required two subprograms are:
   - The subprogram `readPOLY()` which reads and stores a polynomial.
   - The subprogram `writePOLY()` which outputs a polynomial to screen. Due to the fact that it may be hard to do subscripts, the polynomial could be output in a form like this:

     ```
     3x^5 - 2x^3 + x^2 + 4
     ```

2. Create an Ada **package**, `polymath`, which includes subprograms to perform polynomial arithmetic.
   - The subprogram `addpoly(a,b)`, which adds two polynomials together, i.e. `a+b`.
   - The subprogram `subpoly(a,b)`, which subtracts one polynomial from another, i.e. `a-b`.
   - The subprogram `multpoly(a,b)`, which multiplies one polynomial by another, i.e. `a*b`.

3. Create a main "wrapper" program, `poly.adb`, which allows the user to interact with the two packages above, reading in polynomials, manipulating them, and outputting them. Ideally interaction would be menu driven.
   - When a polynomial is input, it should be printed to the screen for the user.
   - When an arithmetic operation is performed, the result should be printed to the screen.

4. Design a good user interface that is intuitive for the user.

NOTES:

- Note that each **package** should include both a `.ads` and a `.adb` file.
- You can add any addition subprograms or packages that you deem necessary for the functioning of the program. For example, you might want to store the data structure for the linked list in a separate module, as it is required by both the main program *and* the packages.

EXTRA TASK

By completely the above task correctly, you have the potential to earn up to **95%** of the grade for this assignment. By going beyond the material given and completely the extra section, you have the potential of earning the remaining 5%.

Add another subprogram, `evalpoly(a,x)`, to evaluate a polynomial, given a specific value for `x` (provided by the user). For example, solve polynomial `a` for `x=7`:

```
(3x² – 5x – 1)
= 3·7² – 5·7 – 1
= 147 – 35 – 1
= 111
```

USER INTERFACE

Here is a **sample** user interface for part of the program, e.g. inputting the polynomial. Your program should use a menu system of some sort (this is up to you to design).

```
POLYNOMIAL Arithmetic
1. Input a polynomial
2. …

Choice? 1
Enter a polynomial:
   Highest exponent: 2
   Coefficients: 3 -5 1
   -> 3x^2 – 5x – 1
Choice?
```

COMPILING

Please do not include a Makefile, and make sure your program compiles in the following manner:

```
> gnatmake -Wall poly.adb
```

ASSIGNMENT INFORMATION

REFLECTION REPORT

Describe your Ada program in one (1) page (single spaced) **reflection report**, explaining decisions you made in the process of designing the program. Consider the design document a synopsis of your programming process. One page should include a synopsis of the approach you took to design the program (e.g. it could be a numbered list showing each step in the process).

Some of the questions that should be answered include:

- Was Ada well suited to solving the problem?
- What particular structures made Ada a good choice?
- Benefits / limitations?

DELIVERABLES

The submission should consist of the following items:

- The reflection report (PDF).
- The code (well documented and styled appropriately of course).
  `poly.adb polymath.adb polylink.adb`

- Both the code and the reflection report can be submitted as a ZIP, TAR, or GZIP file.

STYLING & COMMENTS

Style consists of mnemonic variable names, indentation, and the use of whitespaces and paragraphing. The purpose of good style is to make the meaning of your program clear to someone who has never seen it before, cannot run it, and cannot talk with you. Documentation consists of in-code documentation. Examples of qualities to look for include:

- Are variable names well chosen?
- Are comments relevant rather than simple repetitions of the code?
- Do comments point out key sections of code, indicate special cases, or make assertions?
- Are the indents 3 or 4 spaces? Do not use tabs or 2 space indenting (please check "convert tabs to spaces" in your editor)
- Is whitespacing used to separate parts of the program to provide clarity?

SKILLS

- Ada programming, program comprehension, problem solving