MATHEMATICAL ALGORITHMS I

1. Problem set 2: More multiplication

This set of problems is meant to get more used to information structures. There is some overlap with last week, but you should all be able to getting working algorithims now. We will use multiplication later, so make sure you test everything.

1.1. **Problem: (follow up from last week).** In case you haven't yet implemented classical multiplication of polynomials, then first do so. Make sure the following operations work:

(1) addition, subtraction, and multiplication
(2) variable size: something like `int *coeff; coeff=new int[n];` is okay, but `int coeff[16];` is not for later purposes. The library `vector` is also okay.
(3) implement a simple print function. Input: your favorite data structure for a polynomial. Output: Latex code for the polynomial. For example, the Latex representation of $1 + 5x^7 - x^{18}$ is

```
1+5x^7-x^{18}
```

(4) design and implement an algorithm that differentiates a polynomial. Input: your favorite data structure for a polynomial P Output: your favorite data structure giving the polynomial P'
(5) Same question for integration: in this case you either need to make some data structure for rational numbers or use `float`, `double` or `long double` as a ring.

Combine these operations in a `struct` or `class`. You don't need to do operator overloading (in case you know what this is).

Test your `struct` or `class` with the polynomial $f = 1 + x$. Can you compute $f \cdot f$ and $f \cdot f \cdot f$?

1.2. **Problem: Karatsuba multiplication.**

(1) Implement Karatsuba multiplication of polynomials via an array.
(2) Compare the speed to that of maple or another computer algebra package (for example sympy in python). Can you optimize by using less dynamical memory allocation?

1.3. **Problem: Multivariate polynomials.**

(1) Implement Knuth's algorithm from book I, page 276 for addition and multiplication (also given in the lecture).
(2) Design and implement a print function for multivariate polynomials (represented by a circular list). Input: pointer to the special node of a circular list representing a polynomial. Output: Latex code for the polynomial.