MATHEMATICAL ALGORITHMS I

1. Homework 2:

Homework 2 combines sets 7 and 8 (excluding the resultant computation) It is more work than before. If you cannot manage to do all problems, then hand in a partial solution, such as just multivariate polynomials, or multivariate polynomials and division with remainder, etc.

Again, to make evaluation more convenient: if you work in `C` or `C++`, then we want to see the following rough layout (possibly with some other header files):

```
#include <basic library for in/output>
standard template libraries (if you need them)
a multiprecision library (such as gmp or gmpxx)

Auxiliary functions for polynomials

Division with remainder

S-polynomial

Buchberger

Finding a minimal basis

Reduction


int main()
{
   initialize the ideal by specifying some polynomials

   compute Groebner basis with Buchberger's algorithm
   find reduced basis

   print basis
}
```

If you want to work in a different language, then please ask us. Java is fine (you don't need to ask).

Remark 1.1. Just as last time write your own code (possibly with a partner). There is no point in copying code from the internet. You are supposed to learn something here. Grading will be lenient.

For completeness, the relevant problems from set 7 and 8 are copied here.

1.1. **Multivariate polynomials.**

For the field F use a multiprecision library for rational numbers (I recommend `gmpxx`). The goal of this first problem is to implement multivariate polynomials with linked lists. You can use `list` or `forward_list` from the standard template library, or make your own linked list.

(1) Given two monomials, we can use the lexicographic order to order them, but other orders are also possible. Define a data structure to encode of monomial, and write a function

`lexicographic order` that takes two monomials, say `lhs` and `rhs` (or references, or pointers to such monomials) as input and returns the following output

- 1 if `lhs`>`rhs`
- 0 if `lhs`≥`rhs` and `lhs`≤`rhs` (i.e. if monomials have the same degrees)
- -1 if `lhs`<`rhs`.

The point will be that it should be easy to include other orders later.

(2) Define a class (or struct) to handle multivariate polynomials. The monomials making up the polynomial should be ordered according to a function pointer to the function that returns the monomial order. This can be chosen to be the lexicographic order for now, but your code should be flexible enough to change this to other monomial orders.

Remark 1.2. Other monomial orders include monomial orders that are completely different from the lexicographic order. Make sure you think this step through. This does not just concern the order of the coordinates.

(3) Include a function `insert monomial` that inserts the relevant order in the linked list respecting the order.

(4) Include a boolean function that detects whether the polynomial is zero or not.

(5) Now include the usual operations such as addition, subtract, multiplication (operator overloading is not necessary). Make sure you handle cancellations correctly, i.e. $x - x = 0$

1.2. **Multivariate division with remainder.** Implement the algorithm for multivariate division with remainder.

1.3. **S-polynomial.** Implement a function to compute the S-polynomial of two polynomials.

1.4. **Buchberger's algorithm.** Implement the simple version of Buchberger's algorithm.

1.5. **Finding some minimal basis.** Use the lemma from class to remove unneeded elements from a Gröbner basis. Implement an algorithm performing this task.

1.6. **Finding the minimal reduced basis.** Implement an algorithm that finds the minimal Gröbner basis.

Due May 21st 2018 (until midnight).