CS4 Computer Algebra Syllabus (2018–2019)

As a general rule, the examinable material consists of everything covered in lectures including suggested exercises and related material in the coursework, especially the pencil and paper exercises, or assigned as home reading. More emphasis will be placed on a sound understanding of the main concepts and techniques, and an ability to explain them and apply them to related problems, than on memorizing technical details. However there will be a certain amount 'bookwork' content which can vary between questions, recent past papers are a reasonable guide to this[1]. Below is a summary of topics covered, together with some notes which should be helpful. (The amount of space devoted to a topic is not indicative of its importance.) The summary is organized by sections that correspond to those in the notes.

You will see from time to time that the proofs of certain theorems are non-examinable. You should therefore not try to memorize these but understanding some of the proofs is usually a good way of consolidating your understanding of the basic material of the relevant topic. It is very important that you are able to reason about fairly straightforward situations. For example you should be able to reason about polynomials, appreciate when it is best to view them structurally and take degrees etc., when it might be best to expand them and consider coefficients, or maybe a mixture of both. Likewise you should be able to work with ideals, e.g., if I is an ideal and $f_1, f_2, \ldots, f_r \in I$ then you should (i) know straight away that $(f_1, f_2, \ldots, f_r) \subseteq I$ and (ii) be able to prove this if asked. As another example you should be able to see straight away that, e.g., $2x + 3y \notin (x^2, y^2)$ because every non-zero element of (x^2, y^2) has degree at least 2. You should also be able to see that this reasoning depends heavily on the fact that the generators are power products and fails in general, e.g., $y \in (x^2 + y, x^2)$.

As mentioned above, past exam papers are a reasonable guide but remember that some topics have changed (in particular those covered in the exercises). Note that each examinable topic is equally likely to come up irrespective of what has happened in previous years.

An important point to note is that if you are asked to describe an algorithm you can do it in clear English with appropriate use of Mathematical notation or pseudocode or a mixture. The important point is to give the key ideas clearly. As an example, if you are asked to describe an algorithm for isolating all the real roots of a polynomial then you can discuss the method based on Sturm sequences. The key points are that (i) we make the polynomial square free (state how), (ii) construct the Sturm sequence (so here you must state clearly how that is done), (iii) find an $a > 0$ such that $(-a, a)$ contains all the roots of the polynomial (here it is enough to mention use of either Cauchy bound covered in the notes without further details), (iv) how we use the Sturm sequence to count the roots (v) by repeated subdivision and root counting we can now isolate all the roots. The last part is utterly trivial and does not require any more explanation really. The core is the construction and use of Sturm sequences and that is where most of the marks would be allocated.

§1 Introduction

This consists of the sort of general knowledge expected of somebody who claims to have studied computer algebra.

§2 Brief Introduction to Axiom

You are expected to know about Axiom in reasonable, but not minute, detail (i.e., to the level used in the assignments and as discussed in this section). You could be asked to describe some aspects of Axiom or be given some Axiom code and asked to comment on various points about the code as well as possible further developments. For example you might be asked to enhance given code in some simple ways, justify correctness of some parts (usually by using Mathematical properties of the problem addressed) and/or suggest modifications to improve efficiency (no formal

[1]Any questions involving Maple would now be expressed in terms of Axiom whenever possible, otherwise replaced if I was writing the exam now. In any case there are now several exams based on Axiom and they are a good guide.

analysis). Code might be based on ideas drawn from other parts of the course possibly applied to a simpler situation, e.g., use of modular arithmetic for some straightforward problem such as the multiplication of polynomials or applied to a question relating to their gcd.

You will *not* be required to write large amounts of code—at the very most you might find it useful to include small fragments which need not be 100% syntactically correct. You will also not be expected to remember obscure Axiom functions, only the most common ones. In general a question will describe what Axiom functions do unless they are from the basic set, e.g., `gcd`, `max`. You *are* expected to know the basic programming constructs and the various methods of declaring types or type conversion but minor slips will not attract a heavy penalty (e.g., writing `for v in L do` instead of `for v in L repeat`.

You do need to know how to declare types (two ways for functions) and any implications, e.g., that declaring a parameter of a function ti be of type `UP(x,INT)` means we can only supply integer coefficient polynomials in x for that parameter. The level of knowledeg is required is only what was covered in the notes and exercises along with the feedback provided.

§3 Computer Algebra Systems

Just a bit more general knowledge which will not be required in the exam.

§4 Basic Structures and Algorithms

All the material of this section is examinable except for §4.6.2 *Worst Case Analysis of Euclid's Algorithm*, §4.7.2 *Digression: Formal Power Series* and §4.7.7 *A Remainder Theorem.* Note however that §§4.7.8, 4.7.9 *are* examinable. As mentioned above, it is important that you should be able to reason about polynomials and their properties, e.g., those relating to gcd's. You should also understand, and be able to prove, some of the simple consequences of polynomial division and gcds discussed in lectures. For example you should be able to prove that the quotient and remainder in polynomial division are unique.

§5 Keeping the Data Small: Modular Methods

Make sure you understand the main idea behind the method—see §5.1 *Modular gcd of Polynomials in* $\mathbb{Z}[x]$. Of course you should ensure that you understand the definition of gcd's (e.g., the level of ambiguity and how we can resolve it) as well as the important role of primitive polynomials in this context. On the topic of gcd's you should also be able to reason about them in the general context and relate them to particular ones (over the integers or polynomials). You can omit the proofs of Lemma 5.1 and 5.2 but must understand the significance of the statements. It would be reasonable for a question to be based around a different (fairly simple) algorithm and for you to be asked to prove various facts and/or develop parts of the algorithm.

The statement and solution (together with proof) of the Chinese Remainder Problem for the integers only. Note however that the Chinese Remainder Theorem as such can be stated without any associated algorithm and this was given as Theorem 5.4 in the notes. (So if asked to state the theorem you need only state that and do not need to go into details about any algorithm.) The relevance of the symmetric representation of remainders modulo a given integer; you should be able to prove that if we use an appropriate bound then we can recover the integer if we use a sufficiently large modulus.

You are not expected to memorise the Landau-Mignotte inequality (or the proof) but must be aware of its significance. All of §5.4 is examinable and it would be reasonable to ask you to prove properties of resultants or their main property by a different method (for which you would have guidance and/or hints). In particular you should be able to write down the resultant and be able to use it to develop a new test or straightforward algorithm or use its properties to derive a simple consequence. So far as the algorithm *MODGCD* is concerned concentrate on understanding how and why it works—this is much more important than simply committing it to memory. For example you could be given the results of some steps of the algorithm (possibly together with some extra information) and asked to complete remaining parts. (If there is extra information, not normally available to the algorithm, you should be able to use it sensibly; any extra information would be there to make the calculations shorter.) Omit §5.5 *Modular gcd Algorithm for Multivariate Polynomials.*

§6 Gröbner Bases

Basic definitions: ideals, varieties, significance of change of basis and that the variety remains unchanged (including proof). Hilbert's Basis Theorem (omit the proof) and the Nullstellensatz. The relevance of these to systems of equations and of change of basis. Gröbner bases: intuitive explanation, definition in terms of ideals (i.e., that G is a Gröbner basis for I if and only if for every non-zero $f \in I$ there is a $g \in G$ s.t. $\operatorname{lpp}(g) \mid \operatorname{lpp}(f)$), the basic algorithm *GRÖBNER_BASIS* and concepts needed to understand it (e.g., admissible orderings, $\operatorname{spol}(f, g)$, reduction sequences). You are expected to be able to explain the significance of these concepts, e.g., why are S-polynomials defined in the way given or why do we define admissible orders with the two requirements (and not some others)? The idea of reduction w.r.t. a set of polynomials and how it relates to the ideal generated by the set. Omit Lemmas 6.4, 6.5 and Theorem 6.4. The notions of minimal, reduced and normed Gröbner bases; you will not be asked to reproduce the proofs about these (but as already stated understanding them is one way to reinforce your understanding of Gröbner bases). Note that if asked to explain something from amongst these (e.g., an algorithm for computing Gröbner bases) you do *not* need to explain all subsidiary concepts (S-polynomials, reduction) only the top level notions. Such a question would probably ask you to illustrate your answer with a (provided) simple example and that is where your knowledge of subsidiary concepts would be demonstrated (just by doing things).

You should be able to reason about ideals and take advantage of special situations, e.g., be able to prove that if F is a finite non-empty set of power products then it is a Gröbner basis for the ideal (F). Application to ideal membership and solution of equations with proofs. Relevance of lexicographic ordering to diagonalization of equations and the number of solutions (the significance of elimination ideals). You could be asked to apply these ideas to a similar situation where appreciation of the elimination property is the key. Remember that we can define lexicographic order in a very intuitive way: order the indeterminates then collect together all occurences of indeterminates in a power product and treat as a word in a dictionary. Omit the Mathematical proofs and improvements to the basic algorithm. Omit §6.6, §6.7 and §6.8.

§7 Real Roots of Polynomials

Basic ideas of *isolation* and *approximation.* Relevance of square free polynomials to root approximation. Algorithms for approximation, computing the square free part (but not decomposition) of a polynomial. Overall approach of §7.2 *Real Root Isolation*, relevance of Cauchy's bounds on roots (the statement and proof of Theorem 7.2 are examinable but not those of Theorem 7.3). Sturm sequences: definition (you can omit the proof of correctness) use of sequences. Finding the number of real roots from a Sturm sequence either in an interval $(a, b]$ or the total number by using $V_S(-\infty)$ and $V_S(\infty)$. Note that if asked to give an algorithm for isolating the roots of a polynomial then you need only do that, there is no need to discuss how to shrink the intervals for better approximation (in the past some students have wasted valuable time in this way); see the discussion at the end of the introduction to this document.

The remaining sections (§§7.3, 7.4) are *not* examinable.

Kyriakos Kalorkoti, March 2019