1. Lecture 1 (6.10.11)

We give a glimpse of some of the important notions/buzzwords that will be studied in the course.

1.1. **The notion of divisibility.**

One of the most important notions in (elementary) Number Theory is the one of divisibility. Given a natural number n (i.e. $= 1, 2, 3, \dots$), can we divide them into, say, k parties evenly? Or, more generally, when can one lay out n objects of the same kind and size evenly spaced in a rectangle? Eg for $n = 15$, we can find a rectangle of the form 3×5 in which we can place our 15 objects neatly. In a sense, such a number like 15 can be thought of as occurring also as a "2-dimensional" size. If $n = 17$, say, this is not possible–what about $n = 323$ or $n = 691$?

1.2. **Primes = building blocks for divisibility.**

Numbers for which the latter is not possible (i.e. which only occur as a "1-dimensional size") are called *primes*–we exclude the number 1 here. These play a crucial role in number theory as they are the building blocks of sorts for the set of numbers (multiplicatively speaking, see below).

1.3. **Detecting primeness.**

There seem to be no easy patterns among integers which guarantee primeness, and one can easily misjudge the corresponding nature, i.e. being prime or not, of an integer—even one of the best Number Theorists of his time (Pierre de Fermat 1601–1665) fell into the trap: he saw a "prime pattern" in the following sequence 2, 3, 5, 17, 257, 65537 (all of which are indeed prime) and guessed that this prime property would hold for each sequence member. But about a century later Euler (another NT hero) found a factorisation of the very next number in this sequence.
(Q: which is it? A: $4294967297 = 2^{2^5} + 1$, divisible by 641 and 6700417.)

But at least there are very efficient tests (*primality tests*, of which we will encounter one or two) which check reasonably fast whether a number is prime or not–without having to determine *any* of its prime factors!

1.4. **Fundamental Theorem of Arithmetic: existence and uniqueness of such a decomposition.**

It turns out that each natural number can be written as a product of primes, i.e. we can "decompose" or "factorise" such a number into building blocks. What is more, this decomposition into primes not only exists, there is only one way to do this—the prime factors involved are uniquely determined, as well as how often they occur. This fact is a very basic one, reflected by the name of the corresponding theorem, the "fundamental theorem of arithmetic". One could argue that this is one of the reasons why divisibility is such a powerful notion.

And just to caution you: such a uniqueness of factorization is far from the norm in slightly more general contexts (e.g. in so-called "number rings").

1.5. **Realising such a decomposition.**

How about actually achieving a factorization? Nowadays such a factorisation of a 10-digit number on a computer is instantaneous, even finding the factorization of a number up to 100 digits, say, takes only a few minutes on a standard computer. But even now if we want to achieve the same for a 200-digit number we are often out of luck!

1.6. **Cryptography from discrepancy.** Now suppose we find two huge primes, of about 100 digits each, then it takes a computer an instant to multiply them together but it is in most cases practically impossible for someone else to decompose (of course with the help of a computer) the result back into a product of primes. This amazing incongruence between easy "operation" (here multiplication) and hard "reverse operation" (i.e. factorisation) is actually used in cryptography in everyday life situations like secure data transmission or Internet banking etc.

But in order to understand what is behind this, we first need to get a good grip at the divisibility properties of the integers. On the other hand, we will also need to get a feel for how to encrypt messages and, more importantly, how to encrypt things publicly/in broad daylight (so-called *public key cryptography*) in such a way that only someone with "extra knowledge" can actually decrypt it (without sneaking into the drawer/trash can of the sender, of course).

1.7. **How many primes?** Back to our building blocks: an immediate question is whether there are enough such primes at all; we will soon see that there are in fact infinitely many such. But proofs of this fact are "qualitative" in nature; in fact, so far no-one has ever found a closed formula which produces infinitely many primes, and primes only. Also, we will discover a sensible way to ask "how many" primes there are, which is the content of the famous prime number theorem (an analytic result which we will not prove, stating that the number of primes below a bound x is roughly of size $x/\log(x)$).

1.8. **"Comparing divisibilities".** One further feature of divisibility will be prominent soon: while it can be very hard to decompose a given integer, it is in contrast very fast to "compare divisibilities" of two integers with each other, i.e. finding common divisors if they exist. This goes back 2000 years to Euclid, and the resulting (Euclidean) algorithm is the prototype of a fast and efficient algorithm.

1.9. **Working with congruences.** Divisibility naturally groups integers into classes: two integers a and b are considered indistinguishable (*congruent*) with respect to a chosen natural number n if their difference is divisible by n, i.e. if they leave the same remainder when dividing by n. In this vein, it makes sense to think of all the multiples of n as being zero! It turns out that this notion of being congruent is perfectly compatible with most of the ways we commute with the integers normally, leading to a "calculus of congruences". In fact, we are quite used to thinking with congruences in everyday life, in a sense–"quarter past the hour" can be viewed as the class "15 modulo 60" (if we accept for argument's sake that the time unit in which we measure is a minute). It also shows the ambiguity–if we don't specify the hour or if the context doesn't make it clear, it could indicate one of a whole (ideally infinite) class of possibilities.

1.10. **Quadratic reciprocity of congruences.** One of the highlights of the term will be the exploration of a remarkable structural insight into congruences for the integers which you may not appreciate yet at this stage; to give an example: divide the prime 37 by the prime 11, giving the remainder 4, a square; so 37 leaves the same remainder as a square number when dividing by the 11—we will then say "37 is a square modulo 11". Now the "quadratic reciprocity law" found (i.e. conjectured) by Euler and Legendre and finally proved by Gauss (in six very different ways!) says that one can readily say whether the reciprocal statement "11 is a square modulo

37" holds or not, without actually having to find that square if it exists. (It does: try to divide 14^2 by 37.) We will state the amazing underlying law and give a proof.

1.11. **Other keywords,** which will be explained in due course, comprise e.g. *RSA encryption, the discrete logarithm, primitive roots modulo n* or the *Riemann hypothesis.*

1.12. **Computers.** The use of computers: especially for the cryptographic parts of our course only few examples can be carried out by hand (in decent time); hence it is almost indispensable to rely on calculators (but the range in which they are useful is still rather small) or, better still, on computers. It is very instructive to perform such computational experiments with the notions developed in the course, and for this reason I recommend that you familiarise yourself with one of the following standard packages (in exams, only specific calculators will be allowed, though)

- GP-PARI (free, slim, powerful, focussed on NT);
- Maple (cheap licenses via University, big, not mainly for NT; at times somewhat unintuitive; more powerful for symbolic calculations);
- Mathematica (not so cheap student licenses ($\sim$£100), big, multipurpose; similar ball park as Maple, great graphics routines);
- SAGE (free, huge, very powerful, strong focus on NT; versatile, unifies [i.e., it serves as a shell for other packages like the above, provided license for those is valid]).

[If there is sufficient interest, one lecture may be devoted to give an impression of how to use GP-PARI (in our context).]