The University of Sydney
School of Mathematics and Statistics

Computer Tutorial 1 (Week 1)

MATH2068/2988: Number Theory and Cryptography Semester 2, 2017

Web Page: http://www.maths.usyd.edu.au/u/UG/IM/MATH2068/
Lecturer: Dzmitry Badziahin

The purpose of this first computer tutorial is to introduce you to the mathematics computer laboratories and to the MAGMA computer algebra system. In subsequent tutorials we will use MAGMA to investigate various cryptosystems and number theoretic algorithms.

MAGMA was developed by Prof John Cannon and the Computer Algebra Group at the University of Sydney.

The mathematics laboratory computers that you will use in this unit form a network that is separate from the University's Access Lab network. However, accounts have been created for all students enrolled in MATH2068/2988 with the same usernames and passwords as for the Access Labs. (Note, however, that if you change your password on one network then that change will not automatically flow through to the other.)

The computer you will use is called "Zeno". It should be obvious how to log in. After logging in you should position the cursor anywhere on the Desktop and click the right mouse button. This will bring up a menu of different items that you can use. The most important ones are as follows:

- Logout (down at the bottom). Be sure to log out at the end of your class. (This is separate from exiting from your MAGMA session.)
- Firefox. Clicking this brings up a web browser that will enable you to navigate to the MATH2068 or MATH2988 page and view course material such as tutorial sheets and so forth. You should also be able to access your University of Sydney email account – this will be useful for emailing to yourself a copy of the (automatically created) record of your MAGMA sessions, for the purposes of assignment submission and general revision. Note, however, that most websites outside the University of Sydney are not accessible from the mathematics laboratories.
- MATH2068 Menu (or MATH2988 Menu, which are identical). Pointing at this causes a submenu to appear, in which there is only one item: magma. Clicking this starts the magma program in an environment that is prepared for MATH2068/2988. (If you start magma by typing magma at a command prompt then several special MATH2068/2988 functions will not be available.)
- File Manager. This enables you to see the various files and directories that exist. Initially it will show you the contents of your home directory. The first time you click Magma on the MATH2068 or MATH2988 menu, a subdirectory of your home directory will be created, which will contain the log file of your MAGMA session.

1. Start MAGMA by clicking the MATH2068 or MATH2988 menu item. In a short while the MAGMA program will start and you will be presented with its 'prompt' (>). You can now type MAGMA commands; for example, type

```
print 1 + 1;
```


and observe that MAGMA responds 2. **Every MAGMA command must be terminated with a semicolon; if you enter a command and discover (then or later) that it has no effect, it is probably because you forgot the final semicolon.**

The syntax for other arithmetical operations is what you would expect: for example, to evaluate 4×3, $10 + 20$, and 5^3, at the prompt (>) type

```
print 4*3, 10+20, 5^3;
```

Notice that multiple expressions can be separated with commas; MAGMA will do nothing until you type the final semicolon.

The word "print" was actually superfluous in the above examples: if you just type an expression as a MAGMA command then MAGMA will evaluate the expression and print it. Try it!

Incidentally, if you make a mistake and want to re-type a command, you can use the arrow keys to recall your previous commands to the prompt.

2. Needless to say, MAGMA can deal with numbers far bigger than your calculator can display (or compute) exactly. For example,

```
Factorial(50);
```

evaluates factorial 50 (the product of the numbers from 1 to 50, which mathematicians usually write as 50!), and prints the answer.

MAGMA allows you to assign names (technically called *identifiers*) to expressions. For example, the command

```
x1:=Factorial(50);
```

assigns the value 50! to the identifier `x1`. Now the command `x1;` causes MAGMA to print the value that has been assigned to `x1`.

Identifiers must start with a letter and be followed by any combination of letters and digits. There are a few reserved words that cannot be used as identifiers (`print` is one, `cat` is another); if you try to assign a value to a reserved word MAGMA will complain "bad syntax". Note that MAGMA is case-sensitive: `A` is not the same as `a`.

MAGMA comes equipped with an enormous number of functions that can be used in calculations, such as the `Factorial` function. We will only ever make use of a small handful of these so-called *intrinsic* functions. It is worth noting, however, that almost all MAGMA intrinsic functions have names beginning with capital letters; so it is wise to use only lower case letters in your own identifiers, so that you do not accidentally redefine the name of an intrinsic.

3. We shall occasionally use logarithms. The command

```
Log(100);
```

prints 4.6051701859880913680359829093 7, which is the natural logarithm (logarithm to the base e) of the real number 100, correct to 30 figures. If you wanted the logarithm to

the base 10 then you could get it via `Log(10,100)`. MAGMA accepts any positive real number except 1 as a base for logarithms.

4. When working with classical cryptosystems, we will often need MAGMA to work with *strings*. A string is any sequence of symbols that you can type; a double quote (") indicates the start of a string, and a matching double quote indicates the end. Type

```
loyal:="God save the Queen";
loyal[3],loyal[5..6],loyal[14..18];
```

The first of these commands assigned the value "God save the Queen" to the identifier `loyal`, the second told MAGMA to print the third letter of the string and two substrings.

5. A number theory problem of great importance in cryptography is that of finding efficient ways to factorize positive integers. MAGMA is very good at this. Type

```
Factorization(60);
```

Can you guess what MAGMA's response to your command means? Check by typing

```
2^2*3*5;
```

Use MAGMA to factorize a few other integers, such as 768, 100000020000001, $2^{11}-1$, $2^{2^5}+1$, and confirm the answers by multiplying the factors together (using MAGMA) to retrieve the number that was factorized.

6. MAGMA has an intrinsic function for computing the greatest common divisor (also called highest common factor) of two integers. Test this function by typing

```
GCD(1152,1296);
```

Get MAGMA to randomly choose two integers between 0 and 100000000 by typing

```
a:=Random(100000000);
b:=Random(100000000);
```

and make it tell you what integers it has chosen by typing the commands `a;` and `b;`. Then type

```
GCD(a,b);
```

Repeat the commands `a:=Random(100000000); b:=Random(100000000); GCD(a,b);` as many times as it takes to get a gcd that is not 1. (Remember, you can use the arrow keys to recover commands you have already entered.)

Now let's be more ambitious: try

```
a:=Random(10^10000);
b:=Random(10^10000);
GCD(a,b);
```

Notice how quick this `GCD` function is, even when the two numbers involved are gigantic. You'd better not type `a;` or `b;` at the moment, unless you want to see MAGMA print a number with $10,000$ digits. (Incidentally, if you do accidentally ask MAGMA to do something that would take too long to complete, you can use Ctrl-C to escape.)

7. As we shall see in lectures, there is a simple and efficient procedure for finding the greatest common divisor called the *Euclidean Algorithm*. Enter the following lines of code to define a new function MATH2068GCD which implements this algorithm:

```
MATH2068GCD:=function(a,b)
   while b ne 0 do
      r:= a mod b;
      a:=b;
      b:=r;
   end while;
   return a;
end function;
```

(Notice how the prompt changes to indicate when you are inside a `function` definition or a `while` loop.) After making this definition, you can use the function MATH2068GCD in exactly the same way as the intrinsic function GCD. Test it with

```
a:=Random(10^10000);
b:=Random(10^10000);
MATH2068GCD(a,b);
```

to confirm that the Euclidean Algorithm is pretty good, if not quite as fast as the intrinsic MAGMA function.

***8.** Note that the MAGMA function "a mod b" used above returns the remainder when a is divided by b, also known as the residue of a modulo b. Write a function of your own to compute this, using the knowledge of MAGMA syntax you can get from examining the above definition of MATH2068GCD (or look up the lecture notes). This will be nowhere near as fast as MAGMA's inbuilt function, but still quite fast.

9. To finish your MAGMA session type `exit;` making sure to include the semicolon. Do this now.

Use the File Manager to find the log file for your just completed MAGMA session; it will be in in the magma subdirectory of your home directory. You can read or edit it in any text editor (for example, you may want to delete lines that correspond to typing errors in your magma session). To keep a record for the purposes of revision, you can email the log file to yourself at some other computer account that you use.

Don't forget to log out before you leave!