

## Computer Tutorials: Introduction

The purpose of this tutorial is to introduce you 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.

### Getting Started

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 as for the Access Labs. That is, your username is your unikey. Initially your password will be your SID, but you will have to change it immediately.

At the Windows login screen, login to "ROMEGROUP" using your unikey as username your SID as password. The first time you log in you will immediately get a message saying that your password has expired, and you will be asked to enter a new password. You may as well set your maths password to be the same as the one you use for the Access Labs, thus keeping your maths lab password and your unikey password synchronized. But you can choose a separate maths lab password if you wish.

After logging in you will find several icons on the desktop. There is a shortcut to your "Home" folder: the files you create will be stored here. "Crimson Editor" is useful for editing these files (or use Notepad). There is an "Intermediate Mathematics" web page shortcut (Firefox), and a "Uni web mail" icon. There are also icons named "handin" and "handout" which will be used for electronic submission of assignments. Some of the other icons on the desktop are placed there for the benefit of other units of study, notably statistics, and are not relevant for MATH2068/2988.

Most important for MATH2068/2988 is the icon marked "magma". Double click it. 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

```
1 + 1;
```

and observe that MAGMA responds 2. (Every MAGMA command must be terminated with a semicolon; if you enter a command and nothing happens then it is often because MAGMA is waiting for you to type a semicolon.)

It is recommended that you keep a log file record of each your MAGMA sessions, so that when you want to revise or look back through the things you did in earlier tutorials you will be able to simply look at the logs. To tell MAGMA to start keeping a log, type

```
SetLogFile("week1.txt");
```

at the MAGMA prompt. (The quotation marks are needed; what goes inside the quotation marks is the name you want to give to the log file. It can be whatever you like, but week1.txt would be quite a sensible choice for Week 1. As soon as you type the semicolon a file with the specified name will be created in your Home folder.

After setting a log file you are ready to do the tutorial questions. The questions themselves give you instructions about what to type. To finish a MAGMA session type

```
exit;
```

making sure to include the semicolon.

## Computer Tutorial 1

Log in, start MAGMA, and type a SetLogFile command to ensure that you will have a record of your session.

A *prime number* is a number that has no factors other than itself and 1. Get MAGMA to compute the set of all primes less than 1000:

```
P:={ n: n in [1..1000] | IsPrime(n) };
```

(Do not forget the semicolon! MAGMA will not do anything until you type it.) Now get MAGMA to print out this set:

```
P;
```

It is a basic fact that every integer can be uniquely expressed as a product of prime numbers. Type

```
Factorization(2700);
```

See if you can work out what MAGMA's response to your command means. (Hint: MAGMA is trying to tell you how to express 2700 as a product of prime numbers.) Check by typing

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

Use MAGMA to factorize a few other integers, such as 768, 100000020000001,  $2^{11} - 1$ ,  $2^{25} + 1$ , and confirm the answers by multiplying the factors together (using MAGMA) to retrieve the number that was factorized.

*Fermat's Little Theorem* says that if  $a$  is any integer and  $n$  is prime then  $a^n - a$  is divisible by  $n$ . For example, 7 is prime and  $2^7 - 2 = 126$  is a multiple of 7: it is  $7 \times 18$ . Get MAGMA to choose a random element of the set P defined above,

```
n:=Random(P);
```

get MAGMA to tell you the element it has chosen,

```
n;
```

and then check Fermat's Little Theorem for this value of  $n$  and for  $a = 2, 3, 4, 18$  and 199. For example, use the MAGMA commands

```
b,q:=IsDivisibleBy(199^n-199,n);  
b,q;
```

or simply check that  $(199^n-199)/n$ ; gives an integer answer.

Although you will not have to learn how to write programs in the MAGMA language, you will sometimes be given a few lines written in the MAGMA language and be told to try them out. It will never be difficult to understand such lines, and it is important that you do try to understand them rather than just mindlessly typing the commands. For example, try this:

```
for n in P do  
  IsDivisibleBy(199^n-199,n);  
end for;
```

We have already made use of  $[1..1000]$ , which is MAGMA's notation for the sequence of numbers from 1 to 1000. Sequences can also be specified by writing out all the terms: for example,  $[1,2,3,4,5,6]$  and  $[1..6]$  are equal in MAGMA. If  $s$  is a sequence and  $n$  is a positive integer then  $s[n]$  is the  $n$ -th term of  $s$ . For example

```
seqP:=[ n: n in [1..1000] | IsPrime(n) ];
```

defines  $seqP$  to be the sequence of prime numbers less than 1000, in increasing order, and so then  $seqP[73]$ ; will return the 73rd prime. If you wanted to know the sum of all the prime numbers less than 1000, or their product, you could use  $\&+seqP$ ; or  $\&*seqP$ ; . Try them out!

Besides sets and sequences, we will also make use of *strings*. Strings are defined by enclosing collections of characters inside quotation marks. For example

```
H:="HELLO!";
```

The name you gave your log file was a string: you could have said

```
s:="week1.txt";  
SetLogFile(s);
```

The  $i$ -th character in the string  $H$  is given by  $H[i]$ . Thus, in the examples above,  $s[4]$  is  $k$  and  $H[6]$  is  $!$ . You can use the  $*$  operator to concatenate strings: thus with  $H$  and  $s$  as above,  $H*s$  would be  $HELLO!week1.txt$  Use  $\#$  to get the length of a string:  $\#H$  is 6 and  $\#(s*H)$  is 15.

Each character that you can type has a “code number”. Type the command

```
StringToCode("A");
```

You will find the the code number for A is 65. Conversely, `CodeToString(65)` returns A. Later on we will use code numbers to convert strings into sequences of integers. For example,

```
intseq:=[StringToCode((H*s)[i]): i in [1..#(H*s)]];
intseq;
```

And, of course, the process can be reversed:

```
&*[CodeToString(i): i in intseq];
```

This idea is useful in cryptography, because a message, such as “Meet me at midnight” can be converted into a sequence of numbers and then disguised using arithmetic, in such a way that only the intended recipient knows how to reverse the process.

**A useful tip for these computer classes:** Rather than retyping a long and complicated MAGMA command or sequence of commands given in the tutorial sheet, you can use “cut and paste”. If you do not know how to do this, your tutor, or some fellow student, will be able to show you.

**Another tip:** The up-arrow key will recall previous commands. For example, typing `e` and then pressing the up-arrow key will recall the last command that started with `e`. Note that you do not have to then move the cursor to the end of the line before pressing enter.

**Another:** If you forget the semicolon at the end of a command, then nothing will happen when you press enter. DO NOT RETYPE THE COMMAND! Just type the missing semicolon and press enter again.

Check that the log file for your MAGMA session exists; It will be in your “home” folder. Open the log file using Crimson Editor or Notepad. You may want to edit this file by deleting lines that correspond to typing errors in your magma session, or other such things. If you like, you can email it to yourself at some other computer account that you use.

If you finish everything above with time to spare, here are some more things to play around with.

In the first week of lectures we will discuss the Euclidean Algorithm, which is an extremely efficient method for finding the greatest common divisor (also called highest common factor) of two integers. MAGMA has a built in function for computing the greatest common divisor of two integers. (In fact MAGMA does not use the Euclidean Algorithm but something that is even faster.) Test it 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 `a;` and `b;`. Then type

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

Repeat the three commands `a:=Random(100000000); b:=Random(100000000); GCD(a,b);` as many times as it takes to get a gcd that is not 1. (Use the up arrow key.)

Now let’s be more ambitious: try

```
a:=Random(10^10000);
```

```
b:=Random(10^10000);
```

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

You can define your own functions. Enter the following lines of code to define a function `EuclidGCD`:

```
EuclidGCD:=function(a,b);
```

```
  while b ne 0 do
```

```
    r:= a mod b;
```

```
    a:=b;
```

```
    b:=r;
```

```
  end while;
```

```
  return a;
```

```
end function;
```

(The MAGMA function “ $a \bmod b$ ” used here returns the remainder when  $a$  is divided by  $b$ .)

See if you can figure out what the `EuclidGCD` function does. For example, with a pen and paper try to work out what would be returned if  $a = 21$  and  $b = 15$ .

In fact, the above function is just the Euclidean Algorithm, written in the MAGMA language. Try `a:=Random(10^10000); b:=Random(10^10000); a; b;` and `EuclidGCD(a,b);` to confirm that the Euclidean Algorithm is pretty good.