Chapter 12

Appendix: Maple

(This appendix describes Maple V which is on my machine. Maple 7, which is on your machines, should be backwards compatible.) You can use Maple as a calculator. The nice thing is it does exact arithmetic over $\mathbb{Z}, \mathbb{Q}, \ldots$ For example to calculate $\sqrt{2}/(1 - 3i)$ and then square it, you can type:

```
> sqrt(2)/(1+3*I);

(1/10 - 3/10 I) \sqrt{2}
```

```
> %^2;

-4/25 - 3/25 I
```

Note lines must end with ; (or : to suppress output), % means previous expression, $I$ stands for $i$, and $*$ is the multiplication operator.

You can use variables. Use $\text{:=}$ to assign values.

```
> y := 4;

y := 4
```

```
> (x^2-16)*(x + y)^3;

(x^2 - 16) (x + 4)^3
```

```
> expand(%);

x^5 + 12 x^4 + 32 x^3 - 128 x^2 - 768 x - 1024
```

```
> factor(%);

(x - 4) (x + 4)^4
```
Since \( x \) has no value, the answer will be a polynomial in \( x \). (It’s possible that at some previous stage \( x \) had been given a value. Maple has command \texttt{unassign(’x’) to deal with this.})

Maple has many built in functions. You can also create your own. For simple functions, you can use the \(-\rightarrow\) notation.

\[
Q := (x,y) \to x^2 + y^2;
\]

\[
Q := (x, y) \to x^2 + y^2
\]

\[
N := x \to Q(\text{Re}(x), \text{Im}(x));
\]

\[
N := x \to Q(\Re(x), \Im(x))
\]

\[
N(2-I);
\]

More complicated examples are possible using the constructions “if ... then ... fi”, “for... do... od”, “while ... do ... od”. (This is Maple V; Maple 7 uses a different syntax, but it should still understand the older dialect.) Let’s write our own factorial function in two ways. First we implement the obvious recursion

\[
n! = \begin{cases} 1 & \text{if } n=0 \\ n(n-1)! & \text{otherwise} \end{cases}
\]

\[
faci := n \to \text{if } (n=0) \text{ then } 1 \text{ else } n\text{\textbf{*}}\text{\textbf{f}}(n-1) \text{ fi};
\]

This works; however recursively defined functions are usually not terribly efficient. (We will tend to use recursive definitions in the notes, because it’s often simpler to do so.) As an alternative, let’s compute the factorial by multiplying all the positive integers up to \( n \). Here we use the “\texttt{proc()... end}” syntax. x

\[
fac2 :=
\]

\[
\text{proc}(n) \text{ local } i, f;
\]

\[
f :=1;
\]

\[
\text{for } i \text{ from } 1 \text{ to } n \text{ do}
\]

\[
f := i\text{\textbf{*}}f;
\]

\[
\text{od};
\]

\[
f;
\]

\[
\text{end};
\]

Note that the variable \( i, f \) are declared local since they are internal to the procedure.

\[
faci(6); \text{ fac2}(6);
\]

720

720