Maple Handout

Marius Ionescu, Math 60, Spring 2006

11th April 2006

The following is a handout with (hopefully) useful hints on how to use Maple. They are extracted from the Maple User Guide.

1 Worksheet mode

Maple has two modes: *Document mode* and *Worksheet mode*. The latter is designed for interactive use through Maple commands, which may offer advanced functionality and programming using the powerful Maple language. You can start a new document in Worksheet mode by choosing File->New->Worksheet.

- In Worksheet mode, you enter input at the *Maple input prompt* (>). The default mode for input is Math mode ($2-D$ Math).
- To evaluate an expression (input) press Enter.
- To suppress the output, enter a colon (:) at the end of the input.
- You can also insert input using *Text* mode ($1-D$ Math). This input is in red, and it is entered as a one-dimensional sequence of characters.
- You can press **F5** to switch from $2-D$ Math to $1-D$ Math.
- $1-D$ Math input must end with a semicolon or colon. If you use a semicolon, Maple displays the output. If you use a colon, Maple suppresses the output.
- $1-D$ Math mode is the desired mode for writing programs.

- In both $1-D$ and $2-D$ Math input you can use a semicolon or colon to separate multiple inputs in the same input line.

2 Maple Commands

Commands are contained in the Maple library, which is divided into two groups: *the top-level* commands and *packages*. The top level commands are the most frequently used Maple commands, while packages contain related specialized commands.

- You use a top-level command by entering its name followed by parenthesis: eg: "$>$diff(sin(x),x);" (this differentiates $\sin(x)$ with respect to x).
- In $1-D$ Math input don't forget to include a semicolon or colon at the end of the calling sequence.
- To use a package command, the calling sequence must include the package name, and the command name enclosed in brackets; eg: "$>$randperm[randperm](3);"
- If you use frequently the commands in a package (or to simplify the formulas) load the package using **with**; eg: "$>$with(combinat;ranperm):"; Then you can use the commands as before, eg "$>$randperm(3);"

3 Use Help!

You should always use the Maple Help system when you are looking for an answer (Maple related, of course). The easiest way to get help for a comand is to write **?command**. For example

```
>?for
```

and press **Enter**.

You could also google to search for some already available similar programs.

4 Basic Programming

Two basic programming constructs in Maple are the **if** statement, which controls the conditional execution of statement sequences, and the **for** statement, which controls the repeated execution of a statement sequence.

- The **if** statement has the following syntax:

```
>if conditional_expression1 then
  statement_sequence1
elif conditional_expression2 then
  statement_sequence2
...
else
  statement_sequenceN
end if;
```

The conditional expressions can be any boolean expression. You can construct boolean expressions using

- Relational operators $<, <=, =, >=, >, <>$;
- Logical operators - **and, or, xoe, implies, not.**
- Logical names - **true, false, FAIL.**
- Example of simple **if** statements:

```
>if rand(1..2)() = 1
        then if show then lprint(‘H‘) fi;
             headcounter:=headcounter+1
        else
             if show then lprint(‘T‘) fi;
fi;
```

The **for/from** loop has the following syntax

```
>for counter from initial by increment to final do
    statement_sequence
end do;
```

- The default value for *increment* is 1, for *initial* is 1, and for *final* is **infinity.**
- Example of simple **for** statements

```
>for i to n
   do
      if rand(1..2)() = 1
         then
              if show then lprint(‘H‘) fi;
                      headcounter:=headcounter+1
         else
                   if show then lprint(‘T‘) fi;
      fi;
   od;
```

The **while** loop repeats a statement until a boolean expression does not hold. The **while** loop has the following syntax.

```
>while conditional_expression do
   statement_sequence
end do;
```

- Example of a simple **while** loop:

```
>outcome:=0:
 while outcome<>6 and rolls<>4 do
    outcome:=roll():
    if show then lprint(outcome) fi:
            rolls:=rolls+1:
 od:
```

- You can construct an **infinite loop** (a lope for which there is no exit condition); for example, a **while** loop in which the *conditional_ expresion* always evaluates to **true.** Maple indefinitely executes an infinite loop unless it executes a **break, quit,** or **return** statement.

5 Procedures

A Maple procedure is a program consisting of Maple statements. To define a procedure, enclose a sequence of statements between **proc(...)** and **end proc** statements. In general, you assign a procedure definition to a name.

To improve readability of procedures, it is recommended that you define a procedure using multiple lines, and indent the lines using space characters. To begin a new line (without evaluating the incomplete procedure definition), press **Shift+Enter**. When you have finished entering the procedure, press **Enter** to create the procedure.

- Example of a simple procedure:

```
>with(combinat,permute):
>AllPermutations:=proc(n)
                permute(n);
end:
```

- In recent versions of Maple **with** does not work when called within a procedure.

- To run the procedure **AlPermuations**, enter its name followed by parentheses (( )) and include the parameter:

```
> AllPermutations(3);
```

```
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]
```