UNIVERSITY OF WATERLOO

Maple T.A. Content Developer's Manual

A Guide to Creating Great Questions

by Michael Fattori
of the MFCF Instructional Support Group

Spring 2011

This manual not only discusses how to work within the Maple T.A. programming environment, but also how to create dynamic and robust questions using simple mathematical principles. Always remember that official documents exist with more detailed technical information about the features of Maple T.A. These can be provided by your supervisor.

Table of Contents

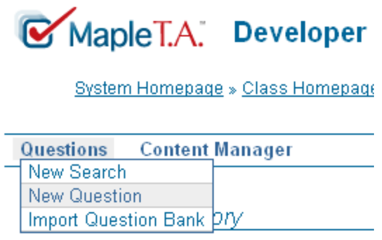
Introduction

This manual contains a variety of tips and tricks that are useful when developing Maple T.A. questions. Readers are advised to become familiar with the Maple T.A. development environment before reading this manual, though a brief discussion of this environment is given below to help get you started. Hopefully this manual will serve as a faster reference than the official Maple documents available to you, but this will be at the expense of the more explicit details about Maple T.A. functionality. However, this document contains tricks (mainly on reverse engineering) that won't be found in the official documentation from Maple, and hence is a valuable resource in its own right.

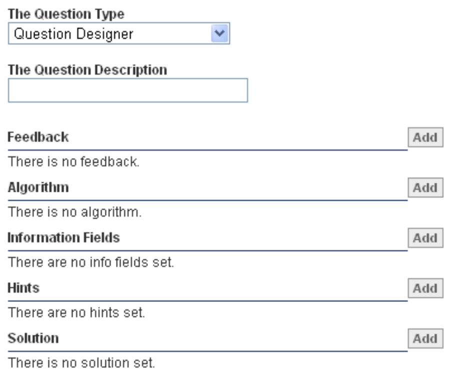
Development on Maple T.A. can sometimes be frustrating, especially when just starting out, but most of the major issues have been dealt with before. This manual should allow you to overcome common difficulties and also help you employ the full power of Maple T.A. to create great questions.

The Maple T.A. Development Environment

You should know how to get to the development website already. This section discusses how to develop a question on that website, not how to navigate it. The first step is creating a new question by clicking the “Questions” button and then clicking “New Question”.



You will be taken to a screen which looks like this

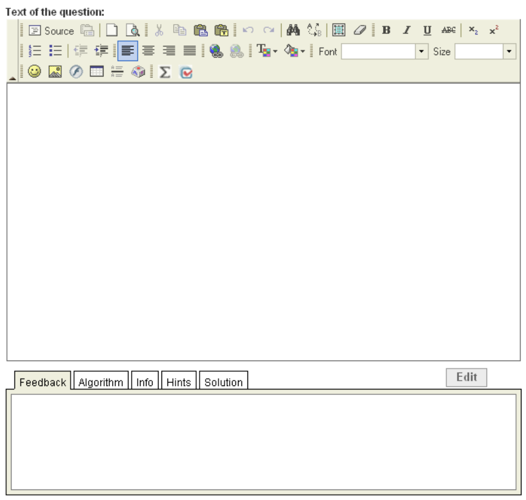

The Question Type
Question Designer

The Question Description

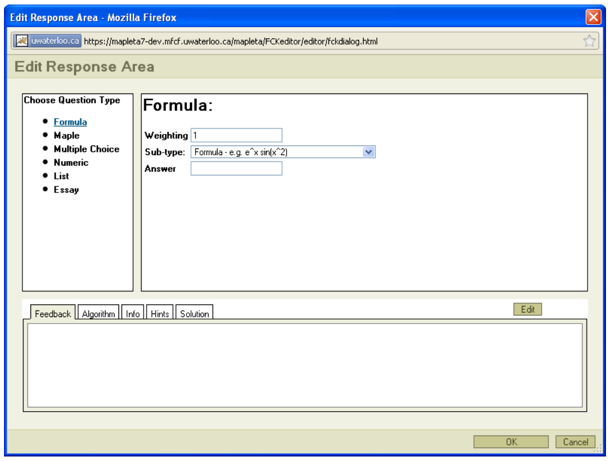
Feedback Add
There is no feedback.
Algorithm Add
There is no algorithm.
Information Fields Add
There are no info fields set.
Hints Add
There are no hints set.
Solution Add
There is no solution set.

We can create most questions using the “Question Designer” type. Enter the name of the question in the ‘Question Description’ field. The most important sections accessible from this page are the feedback and the algorithm. The existing documentation from Maple can tell you about the algorithm editor, namely, how to create variables and what functions are available to you. Just as important to the overall question is the feedback, which is shown to the student after they complete the question, or often after a test is finished. Feedback should be complete, aimed at the appropriate skill level, and hopefully informative to the students.

When the feedback and algorithm are complete you can click “Next” in the upper left corner of the screen. This takes you to a screen which looks like this:



This is where you write the question which students will be given. Variables defined in the algorithm editor can be used here. To give the students a place to enter their answers, place your cursor wherever you want the answer box to appear and click the checkmark symbol from the menu. This leads to a screen where you can edit how you would like the students to be graded.


Edit Response Area - Mozilla Firefox
uwaterloo.ca https://mapleta7-dev.mfcf.uwaterloo.ca/mapleta/FCKeditor/editor/fckdialog.html
Edit Response Area
Choose Question Type
Formula
Maple
Multiple Choice
Numeric
List
Essay
Feedback
Algorithm
Info
Hints
Solution
Edit
OK
Cancel

Formula:

Weighting 1
Sub-type: Formula - e.g. e^x sin(x^2)
Answer

There are several modes for grading.

Formula type: The formula type allows for grading of expressions. If the student's answer should involve variables (e.g., What is the area of a circle with radius r?) then the formula type can be used. This type is lenient on students since it does not require explicit multiplication. That is, $2x$ can be entered as `2x` instead of `2*x`. Use this type for simple expressions of variables. For answers involving functions (e.g., e^x), use the Maple-graded type.

Maple type: We often call this Maple-graded. It allows you to enter Maple code to grade the student's response which is stored as a variable `$RESPONSE`. The answer, defined by you, is stored as the variable `$ANSWER`. This grading type requires a call to Maple on the servers, and so it should only be used if necessary. The last line of the grading code should evaluate to a number between 0 and 1 which represents the mark the student should receive for their answer.

Multiple Choice: This type allows you to develop questions where students must choose one of several possible alternates as the answer. It works fairly intuitively and allows control over where the choices appear. That is, we can make the choices appear in random order or a predefined order, or a mixture of both.

Numeric: This is the simplest type of grading. If the answer is a number, then the numeric grading type can usually handle it. It can be adjusted to accept exact answers or approximate answers. Basic functions (`exp(4), cos(3), sqrt(2)`) and important constants (`Pi, exp(1)`) can be entered as well.

The List and Essay types are not often used by this development team and hence have not been experimented with very often. Feel free to investigate these if the other question types do not seem to suit your purposes. A detailed discussion of all of these question/grading types can be found in the official Maple T.A. documentation.

This document will mostly focus on the feedback and algorithm sections of the development environment because questions are generally not as difficult to grade as they are to algorithmically generate and provide feedback for.

Algorithm Editor

The algorithm editor is where we engineer our questions. We generate random numbers, perform calculations with them, and generate variables for feedback. The functions available to you are found in the manuals from Maplesoft. While some functions are built in to the algorithm editor, we can call on Maple at any time using the `maple()` function. This sends a *string* of Maple commands to our server which uses Maple software to run the commands. Many things you can do in Maple can be accessed using this function. The result of the last line of code executed in Maple is returned to the algorithm editor. If you want more than one thing returned from Maple, return a list of the things you need at the end of the code by separating the items with commas.

Remember that the simplest way to test the call you make to Maple is to run the same code in a Maple worksheet first. With few exceptions, whatever works in a maple worksheet should work from the algorithm editor. Just remember to consider the data type returned from Maple. In particular, if you are asking Maple to return a plot, you need to use the `plotmaple()`

command instead. The `plotmaple()` command is discussed in detail later on in this document.

Feedback

The feedback section is where developers get a chance to educate the students. It allows students to see what they did wrong, and how to do it right. This makes the feedback section one of the most important in question development.

We almost always write the feedback so that it answers the specific question that was asked of the student, and not a generalized version of it. We often develop questions so that the solution can always be found using the same steps. This makes it tempting to provide a generalized solution to the problem, where all numbers have been replaced with variables. However, we try not to do this. It may not be easy for the student to see how the generalized solution can be applied to the specific set of numbers they were given, so we write feedback to help the students who struggle the most with the problem. Always solve the specific problem you give the students so that they can compare their calculations with the feedback and see where they went wrong. Also, we may overlook complications in the calculations needed for a specific problem if we just solve a generalized version of it.

We can use variables defined in the algorithm editor in the feedback section. We often use the equation editor to write the expressions of the solution. There is a separate document on MathML available to you to discuss the details of working with MathML (the language of the equation editor) but most of the work you will do is possible using just the equation editor. To open the equation editor, hit the sigma button Σ from the menu in the feedback section. This opens a window where math expressions can be written just by typing. For instance, x^2 can be written by entering `x^2`. Most expressions can be written intuitively like this. Right-clicking opens a set of palettes that you can use to enter more complicated expressions/functions. Experiment for yourself and read the MathML document for more details.

There are two tabs in the equation editor. The first is where you enter expressions and the other is the MathML tab where we can enter MathML code directly, or edit the MathML code of an existing expression. **When we place algorithmic variables representing numeric values into the equation editor, we almost always need to stop them from being italicized.**

For instance, suppose we define an algorithmic variable $\$x$ to be a random integer. We want to display this integer in the feedback and so we enter `$x` into the equation editor. `$x` will be italicized when we close the equation editor and, when we preview the question, we get an italicized version of whatever random integer `$x` is. Numbers (such as -2 or 6.35) should always be presented as normal text, whereas variables (such as y or f) should be italic. Often you will need to fix this in the equation editor, which requires editing the MathML code. Double-click the `$x` from the feedback section and the equation editor will open up. Click the MathML tab and the code which generates `$x` will be displayed. Look for the algorithmic variable's name, in this case, `$x`. It should appear in the code with tags around it; `<mi>$x</mi>` (search using ctrl-f if you can't find it). To make the algorithmic variable appear normally (non-italicized), enter `mathvariant='normal'` in the opening tag as follows: `<mi mathvariant='normal'>$x</mi>`.

This is what we do for all algorithmic variables which we want to appear as non-italicized numbers, which is most of them. You will be scanning the MathML code in order to make this change often, and should get used to doing so. We understand that this is a bit tedious, and have asked Maplesoft to fix this problem numerous times.

Other than this issue, the feedback is reasonably simple to create. You will find yourself frequently going between the algorithm editor and the feedback section since we often end up defining variables solely for use with the feedback.

Common Issues

This section will discuss the most common development issues, which mainly involve presenting feedback properly. The most common issue was discussed just above, at the end of the feedback section, where we have to change italicized algorithmic variables representing numbers to non-italicized values.

Presenting an Expression

There are a number of things which can go wrong when trying to present an expression to students. The major rules of presenting are:

- Numbers are not italicized
- Variables are italicized
- As often as possible, reduce your expressions to their most succinct forms (except for when not simplifying is more demonstrative to the student).

Fractions

A common issue is to present a fraction in lowest terms. We might want to calculate $\frac{a}{b}$ using the variables $\$a$ and $\$b$. We should present this fraction in lowest terms if we are to use it in further calculations, or if it is a final answer (so almost always). We can make use of two built in functions from the algorithm editor; the `mathml()` function and the `frac()` function. We can define a new variable $\$aoverb$ to be the fraction $\frac{a}{b}$ as follows.

```
$aoverb=frac($a,$b);
```

This is stored as a different data type than other numbers, with a numerator and a denominator in lowest terms, but can still be used in calculations (and grading) as a number. To present this, make a new variable $\$aoverbml$ which will be a MathML expression of the fraction.

```
$aoverbml=mathml("$aoverb");
```

We then place `$aoverbml` wherever we need the fraction in the feedback. Do not place this in the equation editor though since this is already a MathML object. The result would be a broken image .

Sometimes we need the numerator and denominator of a fraction separately so that we *can* place them in the equation editor. This often happens when we need to place the fraction in a larger expression. Unfortunately, placing `$aoverb` into the equation editor will make the fraction appear as a/b instead of $\frac{a}{b}$. To get the numerator and denominator separately, we make the variables

```
$numerator=$a/gcd($a,$b);
$denominator=$b/gcd($a,$b);
```

This separates the numerator and denominator of the fraction in lowest terms. Then the only concern is whether the denominator is ever 1, which we usually don't display in the feedback. If you can reverse engineer your question so that this doesn't happen then the problem is solved (try making the numerator or denominator a prime number), otherwise, more sophisticated manipulation of MathML code may be required

The `mathml()` function will attempt to turn anything you give it into a MathML expression. Try giving the `mathml()` function the exact expression you want to display before working directly with the MathML code. If you must create the MathML code manually for your expression, compose the whole string which will be the MathML code of the expression in the algorithm editor. This is not ideal but it can work.

Fractions have been dealt with frequently by developers, and so ask for help if none of these strategies seem to work. They may have a trick or two up their sleeves.

Polynomials

We frequently work with polynomials but they can be difficult to present. This is best illustrated with an example.

Example

We ask the student, "Find the roots of the polynomial $ax^2 + bx + c$" where we have generated the coefficients a, b and c as algorithmic variables. Suppose, using whatever methods we choose to generate a, b and c, each value can be positive or negative. How do we present this polynomial properly?

A cheap technique would be to present the polynomial as $(a)x^2 + (b)x + (c)$, as in $(-3)x^2 + (2)x + (-5)$. This would allow us to forget about the sign of the coefficients since the expression would always be technically correct. But this is not a properly formatted polynomial. It looks strange, especially if the coefficients are ever 1 or -1. We can do better.

A general way to take care of this is to define two variables for each of the coefficients; one variable to record the absolute value of the coefficient; one variable to record its sign ($+$ or $-$) which should be placed in front of the coefficient. We can also check at this time if the variable has value 1 or -1.

So for the variable $\$a$ we write:

```
$a= … ;
$va=if(eq(abs($a),1), "", abs($a));
$ada=if(gt($a,0), "", "-");
```

The variable $\$va$ is defined as an empty string if the absolute value of $\$a$ is 1, and the absolute value of $\$a$ otherwise. This is because we will place this variable in front of x^2 in the expression and we want x^2 or – x^2 instead of $1x^2$ or $-1x^2$. We might consider forcing the value of $\$a$ to never be 1 or -1 in order to define $\$va$ as simply the absolute value of $\$a$, which will work if the context allows.

We define the variable $\$ada$ to be an empty string if $\$a$ is positive and the negative sign if $\$a$ is negative. Since $\$a$ is at the beginning of the polynomial we would not include a plus sign in front of $\$a$ if it were positive.

For the variable $\$b$ we write:

```
$b= … ;
$vb=if(eq(abs($b),1), "",abs($b));
$adb=if(gt($b,0), "+","&minus;");
```

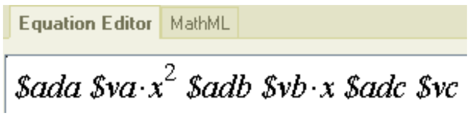
Notice that we define $\$vb$ in the same way as we did with $\$va$.

Also notice the changes made to $\$adb$. Instead of an empty string when $\$b$ is positive, we make $\$adb$ a plus sign. When $\$b$ is negative we make $\$adb$ the “`&minus;`” symbol instead of just “-”. This is because “–” is the small negative sign you would use to indicate a negative number, and “`&minus;`” is the subtraction sign you would use in an expression (it is slightly larger). Note that when you hit refresh in the algorithm editor, the “`&minus;`” will change to “−”. This has not changed it back to the smaller negative symbol, but rather it is displaying the actual “`&minus;`” symbol for you. Unfortunately this means that you must keep track of which subtraction signs are large, and which are small. If in doubt, just erase and rewrite it the way you want.

We define the same extra variables ($\$vc$ and $\$adc$) for the variable $\$c$, except that $\$vc$ should simply be the absolute value of $\$c$ (we want $\$c$ to appear no matter what its value since $\$c$ is not a coefficient).

```
$c= … ;
$vc=abs($b);
$adc=if(gt($b,0), "+","&minus;");
```

Now to write the polynomial, we can enter the following into the equation editor:



$\$ada\ \$va\cdot x^2\ \$adb\ \$vb\cdot x\ \$adc\ \$vc$

Then we can remove the multiplication signs and change every variable to be non-italicized by editing the MathML code. To remove the multiplication signs find
`<mo lspace='0.0em' rspace='0.0em'>&sdot;</mo>`
in the MathML code, occurring after $\$va$ and $\$vb$, and delete it.

The result should be $\mathrm{\$ada\ \$va}x^2\ \mathrm{\$adb\ \$vb}x\ \mathrm{\$adc\ \$vc}$, and the students will see, depending on the values of $\$a, \$b$ and $\$c$, a proper expression like $5x^2 + x - 2$ or $x^2 + 5x + 4$.

Many other presentation issues can be solved using similar techniques as the ones used here. You will likely develop some tricks of your own to save time. This particular problem might also have been solved by making the `mathml()` function do some work. We could have made the first two terms of the expression by having a variable defined as

```
$firstpart=mathml("$a*x^2+$b*x");
```

This would give us the first two terms just fine. Then we would write in the feedback section

$firstpart $\mathrm{\$adc\ \$vc}$

Notice that only the second part is written in the equation editor. We might think that the whole expression could have been written with the `mathml()` function, but if we try to add $\$c$ to the $\$firstpart$ expression (`$firstpart=mathml("$a*x^2+$b*x`**`+$c`**`");`) we get an annoying result, which demonstrates one of many shortcomings of the `mathml()` function in Maple T.A. The result would display as $(6\,x^2 + 7\,x) - 2$ or something similar, with the first two terms bracketed for what can only be understood as absolutely no reason at all.

Note that the sometimes poor functionality of the `mathml()` function is a good reason to become familiar with actual MathML code.

Now, using the techniques of this section, you should be able to handle most presentation issues. One way to avoid presentation issues is to engineer your questions so as to avoid the issues, but that takes time and practice, there is no general method of doing so, and it may not be possible for the particular context of the question. Hopefully in the future Maplesoft will take care of these time-consuming issues so that you do not have to.

Preserving Precision

We often have answers to our questions which are not simple. We frequently want students (especially math students) to enter **exact** answers. This means we need our solutions to be exact so that we compare an exact answer to the student's answer.

π

We frequently have answers involving π (since π is awesome). The trick to preserving accuracy with π is to do all of your calculations involving π on paper. Basically, we get the numbers surrounding π in complete precision so that we never have to resort to storing the expression as a decimal.

Suppose the answer to a question is $\frac{a\pi}{b}$. Then firstly, we should grade the student using the numeric grading-type. In the algorithm editor, we can store the fraction $\frac{a}{b}$ as a variable

```
$ans=frac($a,$b);
```

and then in the answer field for the numeric grading we would write `$ans*Pi`. Of course if we have the variables $\$a$ and $\$b$ we could also write `$a*Pi/$b` in the answer field.

Note the use of capital `P` in `Pi`. Constants in Maple and hence, in Maple T.A. begin with capital letters. Be sure to always use capital `Pi` for π.

We take these measures for preserving accuracy so that we do not calculate an imprecise decimal version of the answer. This would force us to grade the student only to a certain degree of accuracy, whereas the algebraic exact form of the answer is desired.

Just calculate what you need so that the answer field used for grading is the exact expression. More complicated expressions may involve the calculation of more individual parts to the answer.

e^x

There are similar concerns for answers involving e^x. Remember to calculate the parts for the exponent separately and anytime the constant e appears, simply write `exp(1)`.

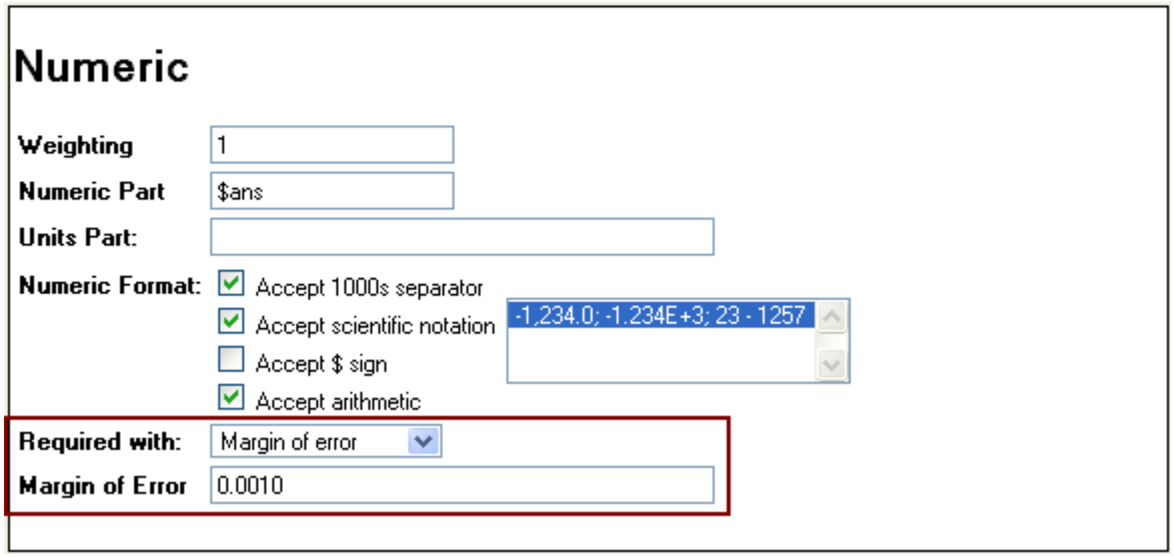
The concerns for other types of exact expressions are the same. In general, try to never use decimals, and find the answer in pieces that can be composed in the answer field.

Decimal Accuracy

Sometimes the mathematics we study involves approximating answers (e.g., many statistics questions). We want students to provide an answer to a certain degree of accuracy, which means students must round their answers to a particular decimal place (specified in the question).

The best approach is to have students round their answer to the nth decimal place by including this instruction in the question, then grading their answer to (n-1) decimal places to avoid any ambiguity in the type of rounding the student or Maple T.A. uses.

Since the answer should be a number anyway, we use the numeric grading type. In the window for editing the numeric answer box, there is an option “Required with:” which has a drop down menu next to it. The default is “Absolute accuracy”, and you need to change this to “Margin of error”.


Numeric

Weighting: 1
Numeric Part: $ans
Units Part:
Numeric Format: [✓] Accept 1000s separator; [✓] Accept scientific notation; [ ] Accept $ sign; [✓] Accept arithmetic — -1,234.0; -1.234E+3; 23 - 1257
Required with: Margin of error
Margin of Error: 0.0010

When you ask students to round to n decimal places, set the margin of error 1 in the $(n-1)$th decimal place. The image above was taken from a question requiring students to round to 4 decimal places, and hence the margin of error was 1 in the 3rd decimal place, i.e. 0.001.

We do this to avoid the discrepancy between the common understanding of “round to the nth decimal place”, and banker’s rounder which Maple T.A. uses. This way, when a student rounds to the specified decimal place the answer will always be correct (assuming their calculations were correct), and any discrepancy is in the student’s favour. This avoids students complaining about incorrect marking since we are grading them on a more lenient order of magnitude than we actually asked of them. Be sure to produce an answer to the required degree of accuracy to compare against the student’s answer.

Reverse Engineering

There are basic concerns when developing a question that you are likely aware of already.

- There must be a solution.
- Any correct solution must be graded as correct.
- Clear language and correct grammar is required.
- The feedback must be correct.

And a number of other concerns could be in this list. It is not just about getting a question to work. We should try to make sure that our questions are not ridiculously hard to solve. A good question only challenges students in the manner intended (mostly). There are many ways that a solution can become complicated and the following tricks should help you reverse engineer questions in order to simplify not only the solution of a question, but the entire construction of a question.

Constructing Polynomials Using Desired Roots

Suppose we must employ the quadratic formula in our solution to find the roots of a quadratic. (This is very common, which is probably why there is a song dedicated to the quadratic formula written by a math teacher http://www.youtube.com/watch?v=D_0vgqi7fU4.) If we must use these roots again in the solution, then we should make sure that those roots are not too complicated to work with. This usually requires us to reverse engineer a question so that roots of the quadratic will always be integers. We illustrate with some examples.

Example 1

The question to the student is, “Find the roots of the polynomial $ax^2 + bx + c$.”

We know that the quadratic formula gives us the answer(s) to this question as

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

We might be tempted to simply define the coefficients a, b and c as random integers. Then all we have to do is define the answer to be both of $\frac{-b \pm \sqrt{b^2-4ac}}{2a}$ and we are done. But this would be an oversimplification. We have a number of things to consider when constructing our quadratic. What if we want there to be two solutions? For instance, x^2 only has one root at 0 (or two equal roots at 0 depending on how you look at it). In fact, any time $b^2 - 4ac = 0$ we have only one root (or again, two equal roots). On that note, whenever $b^2 - 4ac < 0$ we have no real roots.

So it will not do to simply choose random coefficients. A better way to construct this polynomial is to work in reverse and **choose two random roots** and then decide what the quadratic should be.

One solution might be to define two variables, $\$r1$ and $\$r2$ as our roots. If we want to guarantee the roots are distinct we have a number of options:

- We can define them as random integers *in separate ranges*.
- We can define $\$r2$ to be equal to $\$r1$, *plus or minus a constant*.
- We can define the *first as positive* and the *other as negative*.

There are many ways to get the desired result. Be creative and consider the problem you are working on. A good tip is to choose integers as your roots. This means that the solution to the question will be integers, which is often preferable to decimals.

To construct the polynomial with the required roots, remember the factor theorem. If $\$r1$ is a root of a polynomial $f(x)$, then $f(x) = (x - \$r1)g(x)$ for some polynomial $g(x)$.

We can thus construct a quadratic with the roots $\$r1$ and $\$r2$ as

$$\begin{aligned} ax^2 + bx + c &= (x - \$r1)(x - \$r2) \\ &= x^2 + (-\$r2 - \$r1)x + \$r1\$r2 \end{aligned}$$

And we have our values for a, b and c. We even have another degree of freedom in that we can choose to multiply $(x - \$r1)(x - \$r2)$ by any constant we wish. Just make sure to carry out the required calculations to find a, b and c.

And don't stop at quadratics! We can use this method to produce a cubic polynomial with three roots $\$r1, \$r2$ and $\$r3$ by calculating $(x - \$r1)(x - \$r2)(x - \$r3)$, or a quartic polynomial with four roots similarly. Another great advantage of this method is that the coefficients in the expanded polynomial with all be integers when the roots are integers, which is much simpler for students to analyze.

Example 2

The question to the student is "Where does $ax^3 + bx^2 + cx + d$ touch the x-axis and where does it cross the x-axis ?"

Remember that a polynomial will cross the x-axis at roots of odd degree, and merely touch the x-axis at roots of even degree. You will likely only need to worry about roots of degree 1 and 2, but keep this in mind.

Thus to construct our cubic polynomial we can choose $\$r1$ to be where the curve touches the x-axis and $\$r2$ to be where the curve crosses the x-axis. Then our cubic would be: $(x - \$r1)^2(x - \$r2)$. Then we just carry out the calculations as before to determine our coefficients a, b, c and d.

Constructing polynomials from their desired roots can be useful. It helps us have simple solutions, and helps our feedback look cleaner. Sometimes solving for the roots of a polynomial is just one step of a problem. In these scenarios it is much easier to work with integer solutions to do the rest of the problem.

Pythagorean Triples

A Pythagorean Triple is a set of three integers which are the side lengths of a right triangle. These are special since they are integers a, b and c which satisfy the equation $a^2 + b^2 = c^2$.

In some problems, we may find that the solution involves introducing a right triangle and solving for one if its sides. If we want a simple answer from this calculation it helps to be able to engineer the triangle so that it will have all integer side lengths. In other instances we might use Pythagorean Triples behind the scenes in order to guarantee a simple solution to some other problem (an example of this will be provided).

There is a simple formula that gives all the Pythagorean Triples:

Suppose that m and n are two positive integers, with $m < n$.
Then $n^2 - m^2$, $2mn$, and $n^2 + m^2$ is a Pythagorean Triple.

It's easy to check algebraically that the sum of the squares of the first two is the same as the square of the last one. Note that this means **$n^2 + m^2$ is the hypotenuse**.

Here are the first few triples for m and n between 1 and 6.

		$m =$				
		1	2	3	4	5
$n =$	2	[3,4,5]	–			
	3	[8,6,10]	[5,12,13]	–		
	4	[15,8,17]	[12,16,20]	[7,24,25]	–	
	5	[24,10,26]	[21,20,29]	[16,30,34]	[9,40,41]	–
	6	[35,12,37]	[32,24,40]	[27,36,45]	[20,48,52]	[11,60,61]

Example 1

The question to the student is "If $\sin\theta = \frac{a}{b}$, then what is the value of $\cos\theta$?"

We would like to be able to ask this question so that the solution is another simple fraction. If we were to randomly choose a and b then $\cos\theta$ might not be so simple. In particular it may have a radical in the exact expression.

We should first understand how to solve the problem given to the student. If $\sin\theta = \frac{a}{c}$ then we can use the identity $\sin\theta = \frac{opposite}{hypotenuse}$ and then construct the right triangle with angle θ, opposite side length a and hypotenuse length c. Then we calculate the length of the adjacent side b using the Pythagorean Theorem $a^2 + b^2 = c^2$.

In order to ensure b is an integer, the side lengths of the right triangle must be a Pythagorean Triple. Then the answer will be a simple fraction $\cos x = \frac{b}{c}$.

To construct the Pythagorean Triple, we just need to create a suitable m and n, which means m and n are positive integers with $m < n$. The following code accomplishes this:

```
$m=rint(1,4);
$n=$m+rint(1,4);
```

This will restrict m and n to be no greater than 3 and 4 respectively while ensuring $m < n$. We restrict the range of m and n so that the solution will not involve identifying too large of a number as a square. This is important if students are expected to solve these problems without calculators.

Now we construct our side lengths using the formulas, remembering that the hypotenuse must be $n^2 + m^2$.

```
$h=$n^2+$m^2;
$o=$n^2-$m^2;
$a=2*$m*$n;
```

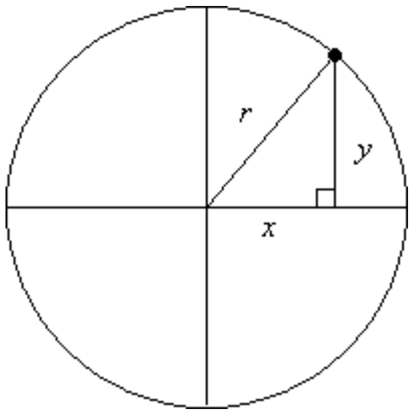
Using these side lengths we can now write the question easily and be confident that the solution will not be too complicated. Thank you Pythagoras!

Example 2

Suppose we wish to ask a student to find the intersection points between a line and a circle centered at the origin.

This sounds fairly straightforward, but there are a number of things we must consider. Since we are generating the line and the circle algorithmically, we must ensure that points of intersection exist. It would also be easier to provide feedback if we knew there were always two points of intersection, which also avoids complications involved with grading. On top of all this, it would be ideal if the points of intersection had integer coordinates so that the solution is not a complicated expression.

An easy way to guarantee that the points of intersection are integers is to create the circle, pick two points on the circle with integer coordinates, then calculate the equation of a line between these points. But does a general circle centered at the origin necessarily pass through points with integer coordinates? Of course, if we guarantee that the circle has an integer radius r, then the points $(r, 0), (0, r), (-r, 0)$ and $(0, -r)$ have integer coordinates, but we do not want to draw lines between only these points since it will not really challenge the student algebraically. For any other points, notice that if the circle has an integer r as a radius, then any points (x, y) on the circle with integer coordinates can be thought to describe a right triangle with hypotenuse r.



Now we see that for any such a point to have integer coordinates (when the radius of the circle is an integer), the side lengths of the right triangle must be a Pythagorean Triple.

So we just need to choose positive integers m and n with $m < n$ so that we can create a Pythagorean Triple. This will determine the radius of the circle $(n^2 + m^2)$, and then we have some flexibility with our choices of x and y.

Choose m and n and calculate the side lengths of the corresponding right triangle as before. Then choose points on the circle. We can pick our first point (x_1, y_1) as follows:

```
$x1=$o*(-1)^rint(0,2);
$y1=$a*(-1)^rint(0,2);
```

This point will lie on the circle (since $\sqrt{x^2 + y^2} = r$). Now we wish to choose another point on the circle so that we may draw a line through both points, but we must make sure that the line has an equation and isn't simply a constant, which means the line cannot be vertical or horizontal.

```
$pick= rint(0,2);
$x2= if(gt($x1,0),
        if(gt($y1,0), switch($pick,0,-$h), switch($pick,0,-$h)),
        if(gt($y1,0), switch($pick,$h,0), switch($pick,$h,0)));
$y2=if(gt($x1,0),
        if(gt($y1,0), switch($pick,-$h,0), switch($pick,$h,0)),
        if(gt($y1,0), switch($pick,0,-$h), switch($pick,0,$h)));
```

This code basically places the second point on the x or y-axis as far from the initial point as possible (while still remaining on the circle). The second point is chosen farther away so that it will be possible to pick a point on the line, interior to the circle, which also has integer coordinates. We did this so that in the question we can describe the line to the students based on this point and the slope (since the y-intercept may not be an integer). It would not be very interesting if we described the line using one of the points of intersection, since then the students would have part of the answer already.

Of course, picking (x_2, y_2) far away from (x_1, y_1) is not enough to guarantee that a point with integer coordinates will lie on the line interior to the circle. We must prove that such a point will exist with the intersection points chosen in order to guarantee that we can always find the interior point. The proof is not too challenging. Essentially, the point interior to the circle will be chosen by moving up/down the rise of the line and left/right the run of the line from either (x_1, y_1) or (x_2, y_2) appropriately (so that we remain on the line and move interior to the circle). The rise and run will be integers, since

$$rise = \frac{y_2 - y_1}{\gcd(y_2 - y_1, x_2 - x_1)} \quad \text{and} \quad run = \frac{x_2 - x_1}{\gcd(y_2 - y_1, x_2 - x_1)}.$$

If the rise and run have a $\gcd > 1$, then the point chosen as above will have integer coordinates (since we are moving from a point with integer coordinates up/down and left/right in integer amounts), and lie interior to the circle. It is a little tricky to prove that the point will lie interior to the circle. The rise and run having a $\gcd > 1$ is what guarantees this (can you explain why?), and once this is understood, the proof only involves showing that the rise and run have a common factor greater than 1. This can be done by rewriting the rise and run in terms of n and m. If we can show this for the two cases when (x_1, y_1) lies in the first quadrant, then the result is true in general by symmetry.

It may not be worth your time to prove this on your own, at least while you are at work. This problem has been written already and exists in the question databases.

Once these key points have been identified though, we can start writing the question, knowing with absolute confidence that no matter what equations need to be solved, the solutions will exist and be simple. The hard work will pay off when we are not struggling to find some way of presenting a solution when it might be an integer or a complicated expression involving radicals, and then needing to use that again later. Not to mention, the final solution will be much simpler for students.

This is far superior to writing static versions of the question using specially chosen numbers. Now a whole classroom could be given this problem and students would not be able to cheat off of each other (at least in the most direct ways). Also, if an instructor wishes, they could raise the bounds on m and n to get even more versions out of the same question without ever worrying that the code will not work.

Reverse Engineering: Final Thoughts

So now we know two great tricks to use when we need to reverse engineer a question: the construction of polynomials based on desired roots; and the use of Pythagorean Triples. Both are powerful tools if used correctly, and act as secure launching pads from which to base questions on. But there is always more work to be done to make the question work correctly, and these tricks are not always applicable.

We always want to make questions as reasonable to solve as possible, which requires an understanding of how the question works. A good way to begin thinking about how a question can be written is to write a static version first. Since many questions are adapted from paper-based questions, we can start by writing a solution to the static question we have on paper. After seeing the steps involved we can try to write our question by ensuring that those same steps can be used when the question is algorithmic. If an expression shows up in the solution that could easily become complicated if the numbers are just a bit different, then that is a good place to start reverse engineering from. Trying designing all the variables involved in the complicated expression so that the expression simplifies easily. For instance, if at some point we rewrote $(\$p - \$q)$ as $(\sqrt{\$p} + \sqrt{\$q})(\sqrt{\$p} - \sqrt{\$q})$, by recognizing that $\$p$ and $\$q$ were perfect squares in a static version of a question, then we should try to guarantee that $\$p$ and $\$q$ are always perfect squares (say, by defining their roots as integers first).

Luckily, the process of writing complete feedback is a natural way to ensure that the question is reasonable to solve. If you are having a lot of trouble writing feedback because there are too many ways that things can change, then you will likely want to engineer the variables involved in order to simplify things. This makes the feedback easier to write and makes the solution more reasonable for students. It’s a win-win situation.

Another example of a question which requires a fair amount of reverse engineering is given below to further illustrate the types of considerations needed when creating a question.

Example

The question to the student is "Calculate the area between the curves $y = \sqrt{k(x+p)}$, $y = -\sqrt{k(x+p)}$ and the line $y = mx + b$."

This question is not so simple. To solve it we would need to integrate with respect to y from the lowest intercept to the highest, if the curves are constructed properly. By that I mean that there is indeed a space between these curves.

We should first note that the two curves $y = \sqrt{k(x+p)}$ and $y = -\sqrt{k(x+p)}$ together form the sideways parabola $x = \frac{y^2}{k} - p$ and so we know that we just want this sideways parabola to intersect the curve $y = mx + b$ in two distinct places. Since we will be using these intercepts as bounds for the eventual integration, we should try to guarantee that the intercepts are at integer values of x and y. We have complete control over all of these variables and so we should be able to do this.

One way would be to choose two points on the sideways parabola with integer coordinates, then draw a line between these two points.

So we just need two points (x, y) satisfying $x = \frac{y^2}{k} - p$ where x and y are integers. We can choose p however we like, so to simplify things, choose p to be an integer.

```
$p=rint(1,6);
```

This will do, though there is some flexibility here. Now we just need $\frac{y^2}{k}$ to be an integer. We haven't even chosen k yet, so let's choose k to be an integer.

```
$k=rint(2,10);
```

Now we can easily make $\frac{y^2}{k}$ an integer by choosing a multiple of k for our first y-value, and another multiple of k for our second y-value.

```
$y1=-$k*rint(1,4);
$y2= $k*rint(4,7);
```

Now we have two points on the sideways parabola with integer coordinates. Since the sideways parabola is symmetric about the x-axis, we chose $\$y1$ below the x-axis and $\$y2$ above the x-axis to make the question interesting. (If you observe the graph, we now have a situation in which integrating with respect to y is simpler than integrating with respect to x, hence justifying our methodology to the student). We made sure that $\$y1$ was not directly below $\$y2$ by choosing to multiply k by a different constant for each, hence giving different x-values. We want this because in the equation $y = mx + b$, m is undefined for vertical lines. If these are the y-values, then the corresponding x-values can be obtained from the equation $x = \frac{y^2}{k} - p$.

```
$x1=($y1^2)/$k-$p;
$x2=($y2^2)/$k-$p;
```

Now that we know we want our line to pass through $(\$x1, \$y1)$ and $(\$x2, \$y2)$ we can calculate m and b for the curve $y = mx + b$.

The slope m can be calculated as $\frac{\$y2-\$y1}{\$x2-\$x1}$:

```
$m=frac($y2-$y1,$x2-$x1);
```

We can substitute one of our points to solve fo r b. Let's use $(\$x1, \$y1)$. We get

$$\$y1 = \frac{\$y2-\$y1}{\$x2-\$x1}\$x1 + b$$
$$\$y1 - \frac{\$y2-\$y1}{\$x2-\$x1}\$x1 = b$$
$$\frac{\$y1(\$x2-\$x1)-\$x1(\$y2-\$y1)}{\$x2-\$x1} = b$$
$$\frac{\$y1\$x2-\$y1\$x1-\$y2\$x1+\$y1\$x1}{\$x2-\$x1} = b$$
$$\frac{\$y1\$x2-\$y2\$x1}{\$x2-\$x1} = b$$

This calculation can and should be done on paper. If we had been content with defining $b = \$y1 - \$m * \$x1$ in the algorithm editor, we would usually get a decimal answer when really, as was shown above, b can be expressed as a fraction. We were also able to cancel out the terms $\$y1\$x1$ and $-\$y1\$x1$ in the second last step, which means that we are able to reduce the computations performed.

It is always a good idea to do paperwork on your own to prevent unnecessary code and reduce calculations made on the server. Often, performing calculations on paper allows us to gain deeper insight as to how questions work, which is important when we are trying to create an algorithmic question from a single, static question, from a paper-based assignment.

Now that we have the equation of our sideways parabola and line which intersect at integer coordinates, we just need to write the question and perform the required integration.

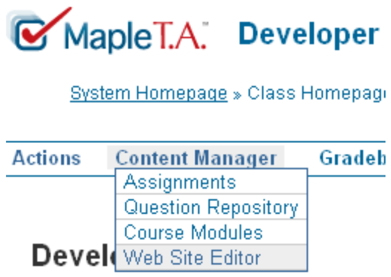
Things You May Need

Here we provide some examples of how to use more advanced techniques in Maple T.A. We will discuss:

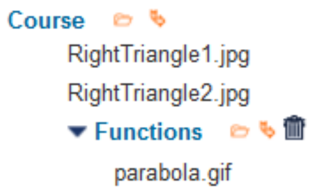
- Uploading images to the web site editor.
- Generating an algorithmic image.
- Grading a list of items entered by the student.
- Using Maple Plots in questions.
- Creating animations for use in feedback.

The Web Site Editor

The Web Site editor can be accessed through the menu dropdowns "Content Manager" and then "Web Site Editor".



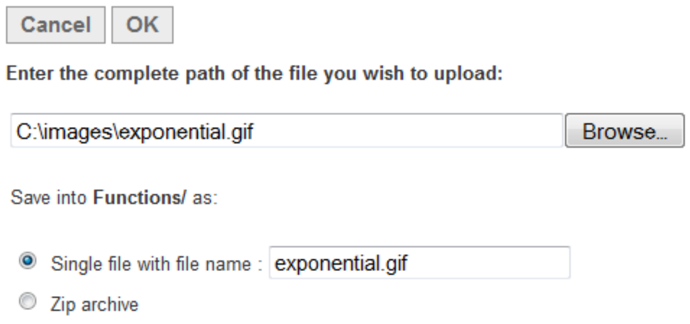
This is where you can upload images and flash movies to use in your questions. You'll be taken to a screen which looks something like this:



The unorganized files at the top are where images that are uploaded using the "Insert/Edit Image" button go. If we upload images from the Web Site editor directly we can choose which folders we want our images to go into. Try to keep your images organized in a logical way.

To upload an image to a folder, say the "Functions" folder from the image above, just hover your mouse next to the folder and hit the "Upload file(s) to this point" button.

This will take you to a screen where you can browse your computer for the image you want to upload and rename the image you are uploading. If you want the same name, just click the entry field for changing the name when you have chosen the image. Otherwise rename it.



Enter the complete path of the file you wish to upload:

C:\images\exponential.gif

Save into **Functions/** as:

● Single file with file name : exponential.gif

○ Zip archive

You can only upload **.jpg** and **.gif** image files. Once the image is uploaded you will be taken to the previous menu. You can look at the pictures in any folder by clicking on the folder name, then selecting the image you want to see.

Movies in **.swf** format (that is, flash movies), can be uploaded in the Web Site editor in exactly the same way, though you will likely not need this as much.

Once an image exists on the server, we can add it to questions using the "Insert/Edit Image" button . We can also use these images to create algorithmic images.

Algorithmic Images

We sometimes want to use an image in a question. We may even want the image to have different numbers or labels appear on it depending on the numbers generated in the algorithm section. We can use images stored on the Maple T.A. server (which you can find in the Web Site editor) for algorithmic images. The following code is the template to really take control of our images, and is placed directly in the source code.

```
<div align=left, right, or center>
<applet code="applets.labelImage.LabelImage" width="width in pixels of your image"
height="height in pixels of your image" codebase="/mapleta/modules">
<param name="image" value="URL of your image">
<param name="size" value="Number of algorithmic labels you want">
<param name="label.1.x" value="x-position of label 1 in pixels">
<param name="label.1.y" value="y-position of label 1 in pixels">
<param name="label.1.text" value="Your label 1">
<param name="label.2.x" value="x-position of label 2 in pixels">
<param name="label.2.y" value="y-position of label 2 in pixels">
<param name="label.2.text" value="Your label 2">
</applet>
</div>
```

The bolded parts are the parts you control. We will look at these from top to bottom.

- `<div align=`**`left right center`**`>`
  First you choose the alignment of the image as one of **left**, **right** or **center**.
- `<applet code="applets.labelImage.LabelImage" width="`**`width in pixels of your image`**`" height="`**`height in pixels of your image`**`" codebase="/mapleta/modules">`
  Find out the width and height of your image in pixels, and enter these in quotations after `width=` and `height=`. You can find these values by opening your image with paint and checking image→attributes in the menu.
- `<param name="image" value="`**`URL of your image`**`">`
  Enter the URL of your image in quotations. You can find the URL of your image in the web site editor.
- `<param name="size" value="`**`Number of algorithmic labels you want`**`">`
  Enter the number of algorithmic “labels” you want in quotations (that is the number of things you wish to algorithmically place on your image).
- The code below is repeated for every algorithmic “label” you wish to place on the image.
  `<param name="label.1.x" value="`**`x-position of label 1 in pixels`**`">`
  `<param name="label.1.y" value="`**`y-position of label 1 in pixels`**`">`
  `<param name="label.1.text" value="`**`Your label 1`**`">`
  Except that for the n^{th} label we write `name="label.n.x"`, `name="label.n.y"`, and `name="label.n.text"`. The x-position and y-position are the xy-coordinates in pixels of the label you wish to place on the picture. The values increase left to right and top to bottom. You will likely need to play around with these values to get things exactly where you intend. Where the code reads **`"Your label 1"`**, enter what you would like to appear in the xy-coordinates you have specified. This can be (mostly) any string, and will work with variables defined in the algorithm editor. We cannot place MathML expressions here (much to the author’s dismay). The most common labels are numbers from the algorithm editor.

Anything that was in bold can have a variable from the algorithm editor go in its place. Just make sure you don’t do something like place the word “hello” as the x-coordinate for your label. Since we have the freedom to include variables from the algorithm editor in this code, we can place any number we want, anywhere we want, and on any picture we want! So we can do far more than simply changing the radius on a circle diagram. We can change the picture itself, and move the labels wherever we need them. Make sure that if you make any such changes, that you make *all* the required changes. If you change the picture, make sure the height and width are changed appropriately as well, and that the labels are in sensible places for the new picture.

To place your image into the feedback or the question itself, enter the above code into the source code directly. If you aren't sure how to read the source code, enter a distinctive string (use "math rules" if you aren't feeling creative) wherever you want the algorithmic image to be. Then enter the source code and hit "ctrl f" (the standard way to search for a string) and search for your distinctive string. Wherever you find the string, replace it with the algorithmic image code.

Grading Lists

Suppose the answer to your question is actually a list of answers. Worse yet, suppose you are not even sure how many items will be in the list! Well, as long as you have a way to generate the answer yourself, you can grade the student's response using the following code:

```
mark:= 0;
res:= [$RESPONSE];
ans:= [$ANSWER];
denomin:= convert(nops(ans),int);
totresponse := convert(nops(res), int);
deduct := max(totresponse-denomin, 0);
i:= 1;
while `not`(nops(res) = 0) do
   if i > nops(ans) then
      res:=subsop(1=NULL, res);
      i:=1;
   elif res[1]=ans[i] then
      mark:= mark+1;
      res:= subsop(1=NULL,res);
      ans:= subsop(i=NULL,ans);
      i:= 1;
   else
      i:= i+1;
   end if;
end do;
evalf((mark-deduct)/denomin);
```

If you understand it, great! Otherwise, th

e code is very robust. Just copy and paste and watch the list grading happen. It even gives part marks! If the student gives too many answers it deducts marks, and only gives marks for correct answers.

This code should be used with the maple-graded type (the only type where you specify grading code), with the maple syntax option. In the answer field, simply enter the answers separated by commas. If the answer comes as a single variable which is a list then just place the variable in the answer field. If this doesn't work, try changing the line

`ans:= [$ANSWER];` to
`ans:= $ANSWER;`

Using Maple Plots

Sometimes we include plots from Maple in our questions. This is good if we need to plot a function that is not always the same in our question.

We can do this from the algorithm editor using the function `plotmaple()`. We use this function in much the same way as we use the `maple()` function, except that `plotmaple()` is specifically used to return images from maple. If you can generate a plot in maple, you can bring it back to the algorithm editor in many cases. The plot will be a variable that you place where you wish in the question.

`plotmaple()` consumes a string, and the string is two arguments. The first argument is the maple code that produces the plot you want. The second argument is for the options of the width and height of the plot returned:

```
plotoptions='width=250 height=250'
```

where the width and height are in pixels. You can change them to be any value, not just 250.

Separate the two arguments with a comma.

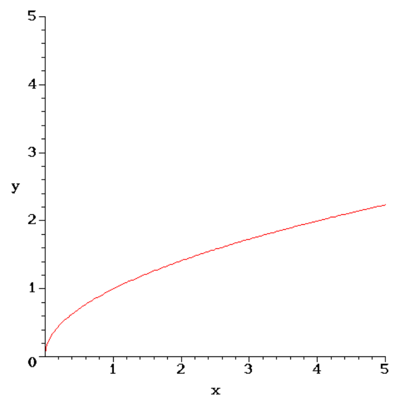
Most plots can be returned from Maple. We will take a look at some examples so as to understand the basics.

Example 1

Suppose we want to ask the student to identify the plot of $y = \sqrt{x}$. A good range to view this function is $0 \leq x \leq 5$ and $0 \leq y \leq 5$. We would enter the following into the algorithm section.

```
$plot=plotmaple("plot(sqrt(x),x=0..5,y=0..5)");
```

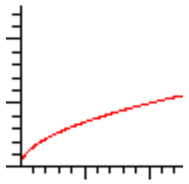
Note that we do not need the second argument (the `plotoptions` argument) to get the plot. A default size is chosen for us. This code would give us the following plot:



To specify a new size, say 100x100 pixels (which is quite small), we would write the following.

```
$plot=plotmaple("plot(sqrt(x),x=0..5,y=0..5), plotoptions='width=100 height=100'");
```

This gives a very small plot:



Usually the smallest plot you will want is 250x250 pixels. Somewhere near 400x400 pixels is the largest. Whatever size you choose, just make sure the important aspects of the plot are visible and clear.

Example 2

Suppose we want to ask the students to identify one of three plots; $y = x^2$, $y = \sqrt{x}$ and $y = \sin\ x$.

If we want a different plot to show up each time, then we could define a variable that switches between the functions we want. Then we just place that variable as the function to plot in the `plotmaple()` call.

```
$function = switch(rint(0,3), 'x^2', 'sqrt(x)', 'sin(x)');
$plot = plotmaple("plot($function, x=-5..5, y=-5..5)");
```

This would give us a plot of whatever the variable $\$function$ happens to be at the time.

While this code would give us the graphs we want, the view of each function is not ideal. The range $-5 \leq x \leq 5$ and $-5 \leq y \leq 5$ is just a sort of "catch all" that will show each function satisfactorily, but is not ideal for any function in particular. If we want a better view of each of these graphs, we need to decide on a range for the view of each function:

For $y = x^2$, let's use the ranges $-3 \leq x \leq 3$ and $0 \leq y \leq 9$.
For $y = \sqrt{x}$, let's use the ranges $0 \leq x \leq 5$ and $0 \leq y \leq 5$.
For $y = \sin x$, let's use the ranges $-2\pi \leq x \leq 2\pi$ and $-2 \leq y \leq 2$.

We could then write our code as follows:

```
$pick3=rint(0,3);
$function=switch($pick3, 'x^2', 'sqrt(x)', 'sin(x)');
$xhi = switch($pick3, 3, 5, 2*Pi);
$xlo = switch($pick3, -3, 0, -2*Pi);
$yhi = switch($pick3, 9, 5, 2);
$ylo = switch($pick3, 0, 0, -2);
$plot = plotmaple("plot($function, x=$xlo..$xhi, y=$ylo..$yhi)");
```

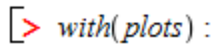
The main idea here is to make variables which are the upper and lower bounds for x and y, then use those variables as the ranges in the `plotmaple()` call. A trick used here is to define a random integer $\$pick3$, then use this as the index for multiple `switch()` calls. This allows us to prepare a set of variables which "go together". This is a good trick in general for questions with multiple versions.

Example 3

Suppose we want to highlight the area between curves on a plot. Say, the area between $\sin(x)$ and $\cos(x)$ on the interval $0 \leq x \leq \pi$.

We need to be familiar with the two functions $\sin(x)$ and $\cos(x)$, in particular, we need to recognize that the two curves intersect at $\frac{\pi}{4}$ on the interval $0 \leq x \leq \pi$. Also, $\cos(x) \geq \sin(x)$ on $0 \leq x \leq \frac{\pi}{4}$, and $\sin(x) \geq \cos(x)$ on $\frac{\pi}{4} < x \leq \pi$.

To highlight the area between the curves we need to make use of some plotting functions from Maple. Firstly, we will need the `display` function from the `plots` package. This function displays several plots on the same graph. It is a good way to compose a plot if no single call to the plot function will give you the entire image you need. In a Maple worksheet, in order to use the `display` function, we can call on the `plots` package by writing:



```
> with(plots) :
```

Then call on the `display` function on a separate line.

Similarly, to use `display` with the `plotmaple()` function we would write:

```
$plot=plotmaple("with(plots):
display(…)");
```

Then we would give the arguments to the `display` function in place of the ellipses.

Note: We can also use the long form of the `display` function which does not require us to call on the `plots` package. This is true of any of the functions in packages that we might use. If you know the function's name and the name of the package it comes from (which can be found in the Maple help files), then the long version of the call to the function is

```
PackageName[FunctionName](function arguments)
```

Now that we know how to call upon the display function, in order to highlight the area between the curves we need to combine a few plots.

The first plot you will need is a plot of the two functions, $\sin(x)$ and $\cos(x)$, on the range we will view the finished plot from. The default range is $-10 \leq x \leq 10$ and y adjusts according to the function. We should view the area on a smaller range, say $-\frac{\pi}{2} \le x \le \frac{3\pi}{2}$ and $-1.5 \le y \le 1.5$. So we can leave the default range since it "covers" the smaller range we will view the finished product from. (This will make more sense when we place all of our plot commands in the `display` function which takes in an argument to specify the view of the combined plots, regardless of the view of the individual plots). We can make a plot of $\sin(x)$ and $\cos(x)$ with the following plot command:

```
plot([sin(x),cos(x)], colour=[red, blue], thickness=2)
```

The thickness is a matter of preference. When filling in areas between curves, the curves themselves may look a bit thin, especially around the filled-in area. Hence the author's preference is to make the function's lines thicker by adding the option `thickness=2`.

Now we need to fill in the area between these curves by making use of the `filled=true` plotting option available through the regular plot command.

The `filled=true` option can be added to a regular plot command. Rather than drawing the curve, the plot command will fill in the area between the curve and the x-axis. The trick to filling in the region between two curves is to fill in the function closest to the x-axis with the colour white, and fill in the function furthest from the x-axis with the colour of your choice (let's say gold). This gives the appearance of only the area between the curves being filled in with colour.

In order for this to work properly, the white fill must come before the colour fill. That is, the function which gives the white filled space must come before the function which gives the colour filled space in the plot command. Also, we must specify the range on which we are "filling", otherwise the fill will be everywhere in the default range $-10 \le x \le 10$.

We can break the areas to fill into two sections: everything above the x-axis; and everything below the x-axis.

Considering the area above the x-axis from $0 \le x \le \pi$, we will need a whitespace under $\sin(x)$ on $0 \le x \le \frac{\pi}{4}$, and under $\cos(x)$ on $\frac{\pi}{4} < x \le \frac{\pi}{2}$ (to see this, the finished plot is shown at the end of this example). If we understand the Heaviside function, we can do this with one plot command, otherwise you will need two separate plot commands to get this. Using the Heaviside function the call is:

```
plot(sin(x)-Heaviside(x-Pi/4)*(sin(x)-cos(x)), x=0..Pi/2,
filled=true, colour=white)
```

To get the gold areas above the x-axis we need to fill in $\cos(x)$ with gold on $0 \le x \le \frac{\pi}{4}$ and $\sin(x)$ with gold on $\frac{\pi}{4} < x \le \pi$. Again taking advantage of the Heaviside function, the call is:

```
plot(cos(x)-Heaviside(x-Pi/4)*(cos(x)-sin(x)), x=0..Pi,
filled=true, colour=gold)
```

Looking below the x-axis, we will need to fill in $\cos(x)$ with gold on the interval $\frac{\pi}{2} \le x \le \pi$. The call is:

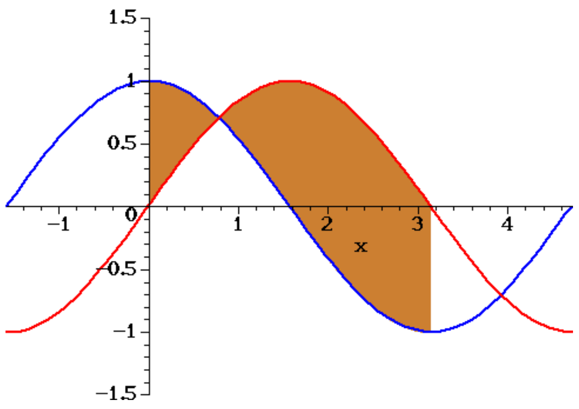
```
plot(cos(x), x=Pi/2..Pi, filled=true, colour=gold)
```

Now that we have the functions and the areas, we just display each of these plots simultaneously using the `display` function. The `display` function simply takes in Maple functions which evaluate to some sort of plot, separated by commas. Then a final argument `view=[a..b, c..d]` is given to view the plots on the range $a \le x \le b$ and $c \le y \le d$.

So we place all of our pieces in the display command to get the final call to `plotmaple()` as:

```
$plot=plotmaple("plots[display](plot([sin(x),cos(x)],
colour=[red, blue], thickness=2), plot(sin(x)-Heaviside(x-
Pi/4)*(sin(x)-cos(x)), x=0..Pi/2, filled=true, colour=white),
plot(cos(x)-Heaviside(x-Pi/4)*(cos(x)-sin(x)), x=0..Pi,
filled=true, colour=gold), plot(cos(x), x=Pi/2..Pi, filled=true,
colour=gold), view=[-Pi/2..3*Pi/2, -1.5..1.5]),
plotoptions='width=600 height=400'");
```

The width and height are just preferences added at the end. The resulting plot from this call is:



Not bad! When you are more comfortable making this sort of complicated call you may be able to algorithmically change the range to be filled in, the functions you are working with, and any of the other aspects of the graph. It usually just requires some basic mathematical analysis of the functions being worked with (but the more you make algorithmic the harder it gets).

Animations

Creating an animation is similar to creating a plot. The underlying idea is to write a plot command with a variable which increases incrementally along a range. Each increment that the variable increases corresponds to a frame of the animation. The easiest way to illustrate how to create an animation is with an example.

Example 1

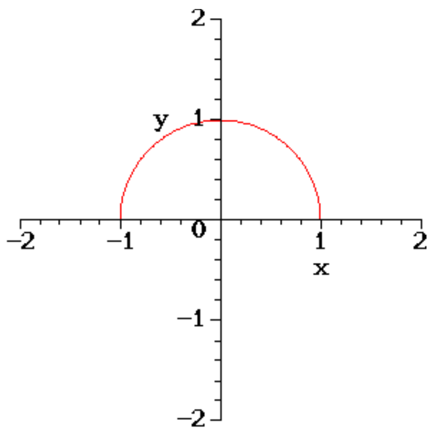
We want to create an animation of the unit circle being traced from $0 \leq \theta \leq \pi$. First we can discuss how to plot using parametric curves. Suppose we just want to see the curve defined by

$$\big(x(\theta), y(\theta)\big) = (\cos(\theta), \sin(\theta)),\ 0 \leq \theta \leq \pi$$

Notice that this is the upper half of the unit circle. To plot this we would use the plot command

```
plot([cos(t), sin(t), t=0..Pi], x=-2..2, y=-2..2)
```

We have replaced θ with t (which is a commonly used variable for parametric curves). If we placed this in the `plotmaple()` function we would get the following graph.



In order to animate the curve being traced out we would use the function `animate()` from the plots package. This consumes three arguments:

- The first is the function you wish to base your animation off of. We wish to animate the plot above which was made using the plot command. So our first argument is `plot`.
- The second argument is a list of the arguments you wish to give to the function specified in the first argument. In our case, the first argument was `plot`, and so the second argument is a list of the arguments we will give to the `plot` command.
- The "third" argument is any number of modifications we wish to make to the animate command. A comprehensive list can be found in the Maple help files, but you will need to

know a few right now. Let's take a look at the code. It is colour coded so that you can distinguish the first, second and third arguments.

```
with(plots):
animate(plot,[[cos(t), sin(t), t=0..A]], A=0..Pi, view=[-2..2, -2..2],
paraminfo=false, frames=50)
```

So we have called on `plot`, and given it the arguments `[cos(t), sin(t), t=0..A]`.

Notice that we did not specify a range to view the animation in the second argument. Instead we specified the view in the "third argument" (in blue) with the call `view=[-2..2, -2..2]` (but we could have specified a range in the second argument if we wanted to).

Also notice that we have made t range from 0 to A. This is how we make the animation work. Each frame of the animation will be the image we would get from the call

```
plot([cos(t), sin(t), t=0..A)]);
```

As A increases from 0 to π this would be an animation of half the unit circle being traced. We specified exactly how A increases in the third argument by writing `A=0..Pi`. The `paraminfo=false` argument is to get rid of a label showing the value of A in each frame of the animation, since we do not normally need to show that information. We can choose to have n frames in the animation by writing `frames=n` (we chose $n = 50$ frames above).

Placing the above two lines of code into the `plotmaple()` function gives the animation. Note that we can choose the variable A to be anything.

Example 2

Suppose we want to show a student how the function $f(x) = a\sin(x)$ changes as a increases. We might decide to range a from -2 to 2, and let's make the curve blue just for fun. Then the call would be:

```
$plot=plotmaple("with(plots):
animate(plot,[a*sin(x), x=-2*Pi..2*Pi, y=-3..3, colour=blue],
a=-2..2, paraminfo=false, frames=50)");
```

Adding too many frames can cause the running time of your question to increase quickly. Around 50 is a good medium quality number of frames, and the default is somewhat less than this. The abilities of the animate function are quite extensive and a more thorough discussion of its capabilities can be found in the Maple help files.

Wrapping Up

Creating great questions takes practice. When you are more comfortable using the techniques explored in this manual you can always experiment and expand on the basics. The best questions are always written with solid mathematical principles working behind the scenes and sometimes creating a problem which is relatively easy to solve can involve math which is well beyond the scope of the solution. Employing these mathematical principles effectively will allow you to write questions more easily, and with greater clarity. Computers are easier to program when solid mathematics is used.

Remember that the feedback you provide students needs to be at their skill level. Employing advanced techniques for a problem may seem obvious to you, but you are not writing the feedback for yourself. If the students understood the advanced techniques, they likely would not need to review the feedback anyway.

Hopefully your work here will help you appreciate the effort which goes into creating effective, educational questions. One of the most difficult tasks is giving the student enough information to find the solution, while not giving too much away. Providing too many hints fails to teach the student problem solving skills. Providing too few hints causes the student excessive frustration. If you can find the right balance you may even learn to read the problems from your own courses more effectively!