Problem Environment

A brief demonstration of the problem environment and nesting interactions for problems with multiple parts.

The problem environment

Basic Facts

The most basic part of a Xronos document is the problem environment. Each individual activity will need to be lead by a `\begin{problem}` and followed by a `\end{problem}`. This is what will handle the layout of the page and the numbering of each of the individual problems. It is entirely possible to have text and page content outside of an environment (E.G. this very paragraph), however life will rapidly become strange if there is an answer box outside of this environment. An example of this would look something like

Problem 1 *An example problem. The answer is, of course, 42.*

$\boxed{42}$

A bit of text after the answer

Which is generated by:

```
\begin{problem}
An example problem. The answer is, of course, 42.

$\answer{42}$

A bit of text after the answer
\end{problem}
```

Note that the code that generates the problem doesn't actually give it a number. This is simply the first `\begin{problem}` and so it is called Problem 1. Each problem environment will cause the page to assign it a number. This behavior will be continued for the rest of the tile.

Nesting

It is a relatively common thing to want to have problems that are related somehow. In such a circumstance you might want to give some thought to how you present that information. Perhaps you want to write a problem but think that the jump from start to end might need a bit of guidance along the way and so you want to ask a relatively simple first question followed by another intermediary step and so on. Perhaps you want to have several preamble problems before having a second stage that combines those pieces of information together. Perhaps you have a problem that has some starting point and then branches out into several different independent subproblems. The answer to each of these concerns is (*clearly*) nesting. By choosing to start a new problem environment before we end one, we create a problem that is only revealed upon completion of the previous. Here are a few examples of how this problem chaining works.

Example 1

First we will look at a problem environment where the nesting has been used to break down each step of the activity. Each answer will reveal the subsequent step in completing our plea for aid.

Problem 2 *In this problem we will spell "help". The first character is* $\boxed{h}$

Learning outcomes:

Problem 2.1 *The second character is* $\boxed{e}$

Problem 2.1.1 *The third character is* $\boxed{l}$

Problem 2.1.1.1 *The final character is* $\boxed{p}$

Problem 2.1.1.1.1 *And now we can escape from this madness.*

Multiple Choice:

(a) *FREEDOM!* ✓

This problem is generated by:

```
\begin{problem}
In this problem we will spell ``help".
The first character is $\answer{h}$
\begin{problem}
The second character is $\answer{e}$
\begin{problem}
The third character is $\answer{l}$
\begin{problem}
The final character is $\answer{p}$
\begin{problem}
And now we can escape from this madness.
\begin{multipleChoice}
\choice[correct]{FREEDOM!}
\end{multipleChoice}
\end{problem}
\end{problem}
\end{problem}
\end{problem}
\end{problem}
```

We have simply started a new "problem" before ending the previous one. This will cause the next problem to be concealed until the correct answer has been put in. Notice how the numbering works for each of the revealed, nested problem environments as we travel through the process. The overall problem is referred to as 2, while each layer appends a .1 to whatever was before it. We can contrast this with something like the following. More information on the "answer" and "multipleChoice" calls can be found later in this document.

Example 2

Here we will help Dark Helmet enter the air shield combination so that we can reveal the punchline.

Problem 3 *The combination is:*

1 $\boxed{1}$ *2* $\boxed{2}$ *3* $\boxed{3}$ *4* $\boxed{4}$ *5* $\boxed{5}$

Problem 3.1 *Which, as everyone knows, is the kind of password that an idiot would put on his luggage.*

Multiple Choice:

(a) *....consider changing the password for your luggage....* ✓

Problem 3.2 *Comedy gold.*

Multiple Choice:

(a) *The works of Mel Brooks are timeless.* ✓

The code that generates that problem is:

```
    \begin{problem}
    The combination is:

    1 $\answer{1}$
    2 $\answer{2}$
    3 $\answer{3}$
    4 $\answer{4}$
    5 $\answer{5}$
\begin{problem}
    Which, as everyone knows, is the kind of password that an idiot would put on his luggage.
    \begin{multipleChoice}
        \choice[correct]{....consider changing the password for your luggage....}
    \end{multipleChoice}
\end{problem}
\begin{problem}
    Comedy gold.
    \begin{multipleChoice}
        \choice[correct]{The works of Mel Brooks are timeless.}
    \end{multipleChoice}
\end{problem}
\end{problem}
```

Note that we don't get the punchline from the fearsome Dark Helmet until after we have entered the correct answer for each of the prompts on that level. Further, the punchline and this author's commentary on that punchline are both different "problems". Each of these subsequent problems could have further nested problems or whole problem sets, but then we might be tempted to return to Problem 2. In particular, the punchline itself is 3.1 and the commentary is 3.2.

This automatic numbering for problems can continue *ad nauseam*. We could go any number of steps deep in this process although it is likely that more than say three or four will cause a revolt among students. You have been warned.

Optional Arguments

The problem environment does not, at the time of this writing, have any optional arguments.

Potential Pitfalls and Problems

Lack of Conditionals

Like all content written at the LaTeX level, this cannot currently be randomized. While you probably don't want to have problems randomly appearing or disappearing at the top level, the catch is that there is no way to make it hinge on a conditional. For example, it might be desirable for a nested question to show up after a certain number of incorrect responses on the part of the student, but at this time no such functionality exists.