

# A User's Guide to the UC Exam Class

Michael JasonSmith

7th December 2004

## Abstract

This guide will show you how to use the UC Exam class to create written exams and tests. If you are new to L<sup>A</sup>T<sub>E</sub>X or wish to alter how the UC Exam class functions, then you should read the ‘The Not So Short Guide to L<sup>A</sup>T<sub>E</sub>X2e’ or ‘A Hacker’s Guide to the UC Exam Class’ respectively.

## 1 Introduction

The UC Exam class typesets assessment items, such as exams or tests, and checks that the marks add up correctly (Section 2). It also has the ability to typeset exams which include space for students to write answers, in addition to the typeset questions (Section 3).

## 2 A Standard Exam

A standard exam is one that just contains questions. A student writes his or her answers in the standard University Answer Book. The answer books are collected at the end of the exam and marked.

To write a question-book, first specify some generic information about the assessment item (Section 2.1), before writing each question (Section 2.2). If there are any problems with the marks allocated to the questions, the class will generate warnings, telling you about the problem (Section 2.3).

### 2.1 Declaring an Exam

To start an exam, declare that you are using the `ucexam` document class, and renew the `\UCPaperCode` and `\UCPaperTitle` macros. For example, the following is the start of an exam for COSC208.

---

```
\documentclass{ucexam}

\renewcommand{\UCPaperCode}{COSC208, ELEC208}
\renewcommand{\UCPaperTitle}{C Programming}
```

---

The first line declares that the UC Exam class will be used. The next two lines set the value of the course-codes and the title. As can be seen in the example, there may be multiple codes, but there should only be *one* title.

By default the UC Exam class states on the title page that the assessment item is a mid-year or end of year assessment item, automatically determining which is appropriate. To change this wording, renew the `\UCSemester` macro, as follows.

---

```
\renewcommand{\UCSemester}{Summer}
```

---

However, you should rarely have to renew the `UCSemester` macro.

If the assessment item is *not* an exam, then the `\UCAssessmentType` macro should be altered to reflect this. For example, the following code changes the assessment from an exam to a lab test.

---

```
\renewcommand{\UCAssessmentType}{Lab Tests}
```

---

The assessment type should always be a *plural*, or the wording on the title-page would be incorrect!

The time taken to complete the assessment item should also be declared in the preamble of the document. By default, the time is set to `THREE hours`, but this may be altered by renewing the `\UCExamTime` macro. The following example changes the time to complete the exam to two hours.

---

```
\renewcommand{\UCExamTime}{TWO hours}
```

---

There is no requirement for the scalar-component of the time to be written in majuscule letters, but tradition dictates that this is so.

The instructions given to the students are declared next. By default the exam is:

- Closed book, and
- Calculators are not allowed.

However, either of these parameters can be altered.

If the exam should be open-book, then set the value of the `\UCBook` macro to `open`, as follows.

---

```
\renewcommand{\UCBook}{open}
```

---

Alternatively, you can set `\UCBook` to an arbitrary string, which will appear in the instructions as-is. The following is the instruction given to COSC326 students for their end of year exam.

---

```
\renewcommand{\UCBook}{Two A4 pages (two sides of an  
A4 sheet of paper) of \emph{hand-written} notes  
may be used.}
```

---

To allow calculators, call the `\UCCalculators` macro, with the argument `true`, as follows.

---

```
\UCCalculators{true}
```

---

Calculators are not allowed by default.

## 2.2 Questions

The questions for the assessment item appear within a `Questions` environment, which appears within an `Exam` environment, which is placed within the body of the document. The following is an example of a very short exam.

---

```
\begin{document}  
\begin{Exam}[15]  
  \begin{Questions}
```

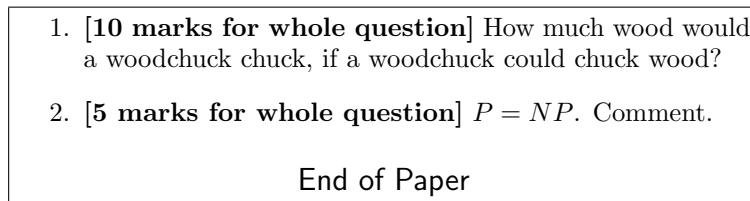


Figure 1: How the questions in a very short exam would be typeset.

```

\Question[10] How much wood would a woodchuck
chuck, if a woodchuck could chuck wood?

\Question[5] $P=NP$. Comment.
\end{Questions}
\end{Exam}
\end{document}

```

---

Things to note.

- The `exam` environment has one optional argument: the total number of marks for the assessment item. If omitted, the assessment item is presumed to be out of 100.
- Each question appears within a `Questions` environment, which can be nested three deep (to create questions, subquestions, and sub-subquestions).
- Each question has an optional argument: the number of marks assigned to each question (defaulting to zero).

Figure 1 shows how the questions in the above example would be typeset.

Questions can be nested, as follows.

---

```

\Question[15] Consider a woodchuck that is
chucking wood.
\begin{Questions}
\Question[5] If working alone, how much wood
would the woodchuck chuck?
\Question[5] If two woodchucks were chucking
wood, would the woodchucks chuck more wood?
\Question[5] Discuss the relationship between
the number of woodchucks available to chuck
wood, and the amount of wood that is
chucked.
You should note
\begin{Questions}
\Question Whether the relationship is linear,
and
\Question If there is a maximum number of
woodchucks that can chuck wood.
\end{Questions}
\end{Questions}

```

---

In this example, the inner-most questions have no marks assigned to them, and are provided as guidance to the student, as can be seen in Figure 2.

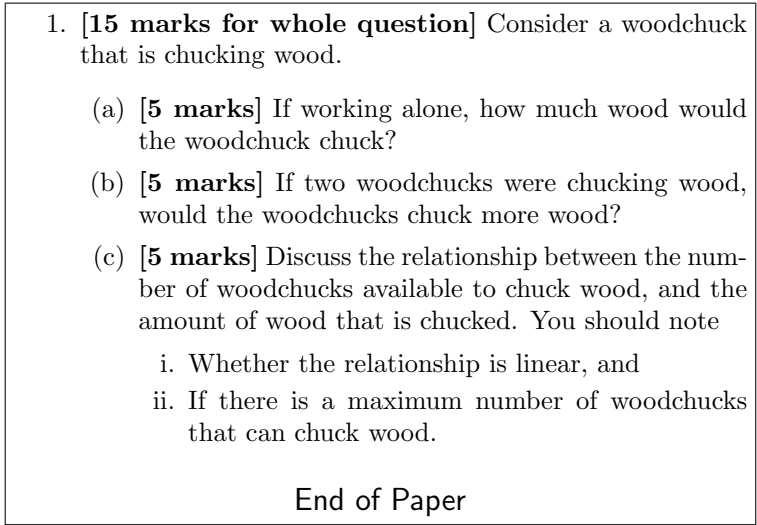


Figure 2: How the nested questions are typeset.

### 2.3 Checking the Marks

As well as typesetting the exam, the `UC Exam` class also checks to see if:

- The marks allocated to all the questions add up to the marks allocated to the entire exam,
- No subquestion (or sub-subquestion) is worth more than its parent question, and
- That the total number of marks allocated to the subquestions (or sub-subquestions) equals the marks allocated to the parent question.

If any of these restrictions are violated, the `UC Exam` class will display an error when `LATEX` is run. In addition to displaying the error, it is written into the `LATEX log file`. The log file is named after the original `.tex` file, but with the extension replaced with `.log`.

The following is an example of the type of error produced by the `UC Exam` class.

```
***EXAM: W A R N I N G
***EXAM: Question 1.a is worth 3 but the sub-subquestions are worth 2.
***EXAM: W A R N I N G
***EXAM: Question 1 is worth 12 however Question 1.b is worth 23
***EXAM: W A R N I N G
***EXAM: Question 1 is worth 12 but the subquestions are worth 26.
```

Each line is prefixed with `***EXAM:`, so you can retrieve all the erroneous marks by using `grep` on the command-line. For example, the following displays all the erroneous marks for the exam named `my_exam.tex`, which are stored in `my_exam.log`.

```
grep "***EXAM:" my_exam.log
```

## 3 A Question and Answer Exam

A question and answer exam is one in which there is no separate answer book. Instead, space is provided for answers to be written in the same book that contains the questions.

For the most part a question and answer exam is the same as a standard UC Exam, but a few extra macros are used. After the exam is declared (Section 3.1) you then specify the questions and the space for the answers (Section 3.2).

### 3.1 Declaring an Exam

The declaration for a question and answer exam is similar to the declaration for a standard exam (Section 2.1) except the `ucexamqa` class is used. For example, the following is a declaration for a question and answer exam.

---

```
\documentclass{ucexamqa}

\renewcommand{\UCPaperCode}{COSC208, ELEC208}
\renewcommand{\UCPaperTitle}{C Programming}
```

---

The other declarations, for the time allowed, whether the exam is open or closed book, and whether calculators are allowed are all the same as for the standard UC Exam class.

### 3.2 Questions and Answers

The `Exam` and `Questions` environments are the same as the standard exam. However, there are three new commands related to providing spaces for students to write answers into:

- `Answer` (Section 3.2.1),
- `AnswerBox` (Section 3.2.2), and
- `AnswerBlock` (Section 3.2.3),.

#### 3.2.1 Answer

An `Answer` creates a lined box into which a student can write an answer. The macro takes a single argument: the height of the box in *lines*. Each line is 7mm apart by default (the same gap as the lines on a pad of refill paper), but this can be changed by calling the `UCAnswerLineHeight` macro and passing the new height as an argument.

In the following example the student is given one line to answer the first question, but 5 lines to answer the second.

---

```
\begin{document}
\begin{Exam}[15]
  \begin{Questions}
    \Question[10] How much wood would a
      woodchuck chuck, if a woodchuck could
      chuck wood?
    \Answer{1}

    \Question[5] $P=NP$. Comment.
    \Answer{5}

  \end{Questions}
\end{Exam}
```

1. **[10 marks for whole question]** How much wood would a woodchuck chuck, if a woodchuck could chuck wood?

---



---

2. **[5 marks for whole question]**  $P = NP$ . Comment.

---



---



---



---



---



---

End of Paper

Figure 3: How the `Answer` macro is typeset.

---

```
\end{document}
```

---

Figure 3 shows how the above code would be typeset.

### 3.2.2 Answer Box

An `AnswerBox` is used when an *unlined* space is needed for the student's answer. The single argument specifies the height of the box as a *as a decimal fraction* of the page height. For example, the following creates a box for the student's answer that is quarter the height of the page.

---

```
\Question[5] Plot the relationship between the
      number of woodchucks available to chuck
      wood, and the amount of wood that is
      chucked.
\AnswerBox{0.25}
```

---

Figure 4 shows how the above code would be typeset.

### 3.2.3 Answer Block

The final, and most complex, of the three commands that generate space for answers is `AnswerBlock`. It allows you to mix typeset text with lines that the student can write on. `AnswerBlock` is quite different to `Answer` and `AnswerBox` because it is an environment and the size is not specified by an argument. Instead, the macros contained within the environment specify the height of the `AnswerBlock`.

The following example uses all the features of an `AnswerBlock`.

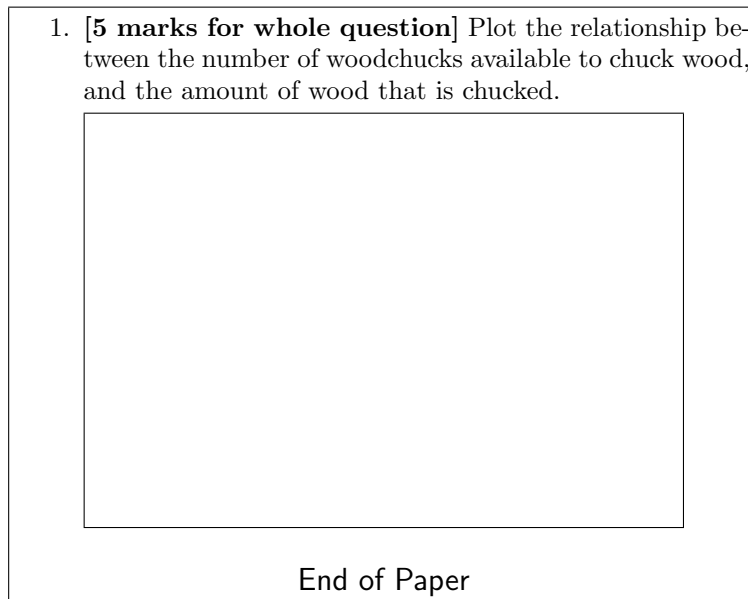


Figure 4: How the `AnswerBox` macro is typeset.

---

```

\Question[5] Write a function to swap the
              values of two integers that are
              pointed to by the arguments to the
              function.
\begin{AnswerBlock}
  \TextLine{\lstinline+void swap(+)\\
  \lstinline+{+\\
  \AnswerLine[4]\\
  \lstinline+}+
\end{AnswerBlock}

```

---

Figure 5 shows how the above code is typeset.

A `TextLine` is used to typeset a line that contains both text that you provide, and text that the student writes. The text that you provide is given as an argument to the `TextLine` macro. In the above example, the start of a function is provided as a hint to the student.

The `AnswerLine` macro is used to typeset a ruled line on which the student can write. By default it will only create a single line, but more lines can be typeset by specifying the number required as the optional argument, as shown in the above example.

Any lines that you do not want the student to write on should contain standard  $\text{\LaTeX}$  macros. In the above example, the opening and closing braces are typeset using the `lstinline` macro, which is available from the standard `listings` package.

1. **[5 marks for whole question]** Write a function to swap the values of two integers that are pointed to by the arguments to the function.

```
void swap(
```

{

}

End of Paper

Figure 5: How the `AnswerBlock` environment and the `TextLine` and `AnswerLine` macros are typeset.