The `brandeis-problemset` Document Class

Rebecca Turner*

2018-10-19

Abstract

Brandeis University's computer science ("COSI") courses often assign "problem sets" which require fairly rigorous formatting. The `brandeis-problemset` document class, which extends `article`, provides a simple way to typeset these problem sets in LaTeX.

Although `brandeis-problemset` is compatible with all LaTeX flavors, XeLaTeX or LuaTeX is recommended for `fontspec` support.

Note: The `brandeis-problemset` document class should be considered experimental; the only stable API is that of the `problem` environment.

Note: Browse the sources, contribute, or complain at `github.com/9999years/brandeis-problemset`

Contents

*Brandeis University; `rebeccaturner@brandeis.edu`

1 Default behavior

`brandeis-problemset` provides packages and well-formatted constructs (notably the `problem` environment) for problem-set writers. `brandeis-problemset` will always render its body copy as a Times variant (`stix` for plain LaTeX or `xits` with XeLaTeX or LuaTeX) and always contains a useful header (which contains the page number, author's name, course, instructor, and assignment due date).

1.1 Default packages loaded

1. `hyperref`, for a nicely-linked table of contents; `\href{url}{label}`.
2. `listings`, for verbatim code listings (including the `assembly`, `java`, and `pseudocode` environments).
3. `xcolor`, for gray line numbers in code listings (and perhaps colored listings in the future); e.g. `\color{gray}`.
4. `enumitem` for better control over the margins and spacing of the `enumerate`, `itemize`, and `description` environments.
5. Math packages:
   (a) `amsmath` for tons of useful math commands, including `\text`, `\intertext`, and `\boxed` as well as the `bmatrix`, `multiline`, `gather`, `align`, and `alignat` environments. See "User's Guide for the `amsmath` Package" for a more complete listing.
   (b) `mathtools` for other useful/utilitarian commands.
6. Table packages:
   (a) `multirow` for cells spanning multiple rows.
   (b) `booktabs` for good-by-default tables and the `\cline` macro.
   (c) `tabu`, the best table package with dynamically resizable columns, easy creation of new column types, and more.

2 Class configuration

2.1 Class options

`brandeis-problemset` defines a limited set of key-value options that may be set at `\documentclass`-time. These may be removed entirely in a future release, as it seems "messy" to have three configuration methods (`\documentclass` options, `\problemsetsetup`, and singular option commands).

`duedate`	Assignment due date in full
`instructor`	Instructor name in full
`course`	Course name in full
`assignment`	Assignment name in full
`gantt`	Load packages for the `ganttschedule` environment
`antonella`	Use Dr. Antonella DiLillio's preferred styles (Courier for code)

Given that `\documentclass` option parsing is much more limited than other key-value interfaces, these options have limited capabilities.

2.2 Setting options after loading `brandeis-problemset`

`\problemsetsetup` {⟨*options*⟩}
Sets global `brandeis-problemset` options; see table 1 for a list of valid options.

Table 1: Options for `\problemsetsetup`; many of these are just used in document headers.

`course`	Course name in full.
`coursenumber`	Course name shorthand; use `21a` for "COSI 21a".
`assignment`	Assignment name in full.
`number`	Assignment name shorthand; use `3` for "Problem Set 3".
`duedate`	Due date, e.g. `2018-10-18`; not parsed at all.
`instructor`	Course instructor.
`codefont`	With XeLaTeX or LuaTeX, pass the given font to `\setmonofont` and enable Unicode shortcuts for the `pseudocode` environment. (If you need to specify options to `\setmonofont`, use `\setcodefont`.)
`gantt`	Load packages for the `ganttschedule` environment
`antonella`	True/false (default: false; if no value specified, assumes true); use Dr. Antonella DiLillio's preferred styles (Courier for code)

`brandeis-problemset` additionally provides a number of configuration commands with similar interfaces as the TeX macros `\author`, `\title`, and `\date`.

`\duedate` {⟨*date*⟩} sets the due date in full.

`\instructor` {⟨*name*⟩} sets the instructor name.

`\course` {⟨*name*⟩} sets the course name in full.

`\coursenumber` `{`⟨*number*⟩`}` sets the course name by number; e.g. `\coursenumber{21a}` gives a course of "COSI 21a".

`\assignment` `{`⟨*name*⟩`}` sets the assignment name in full.

`\problemsetnumber` `{`⟨*number*⟩`}` sets the assignment name by number; e.g. `\problemsetnumber{3}` gives an assignment of "Problem Set 3".

`\setcodefont` `[`⟨*fontspec options*⟩`]{`⟨*font name*⟩`}`
Sets the monospaced font to ⟨*font name*⟩ and uses it for shortcuts in the `pseudocode` environment.

2.3 Practical usage

You may find it useful to define a `.sty` file for each course. For instance, `cosi21a.sty` might read:

```
% set course/author data
\problemsetsetup{
  instructor=Dr.\ Antonella DiLillio,
  coursenumber=21a,
}
\author{Rebecca Turner}

% get a prettier code font -- these can be pretty big so they're not loaded
% by default
\setcodefont[
  Extension = .otf,
  UprightFont = *-Regular,
  BoldFont = *-Bold,
]{FiraMono}
```

and then `ps1.tex` might read:

```
\documentclass{problemset}
\usepackage{cosi21a}
\problemsetnumber{1}
\duedate{2018-10-29}
\begin{document}
% etc.
\end{document}
```

See section 4 for a more complete example.

3 User commands and environments

`brandeis-problemset` provides a number of commands for typesetting problems.

`problem` [⟨*options*⟩]

Defines a problem. A problem is set 1 inch from the left margin (although this amount may be customized by modifying the `\problemindent` length) and begins a new page.

`title`	A problem title, to be displayed after "Problem" and the problem's number.
`number`	A problem number; if given, the problem-number counter will not advance. The number must be robust, because it goes inside a `\section`.
`pagebreak`	True/false (default: true). Add a pagebreak before the problem?

Vertical material is allowed in a `problem`.

`\subproblem` [⟨*description*⟩]

Prints a sub-problem, i.e. a `\subsection`. It doesn't do very much at the moment.

`\Th` [⟨*colspec*⟩]{⟨*header*⟩}

Prints a table-header in bold. By default, the header is left-aligned, but arbitrary alignments can be specified with ⟨*colspec*⟩. `\Th` is backed by `\multicolumn`.

`pseudocode` [⟨*keywords*⟩]

Prints pseudocode.[1] Several shortcuts are defined, as shown in table 2.

These shortcuts display in `\pseudocodesymbolfont` (default: `\ttfamily`), which may be redefined if you prefer something else. The easiest way to change `\pseudocodesymbolfont` is with `\setcodefont`. If you use the `antonella` option with XeLaTeX or LuaTeX, `brandeis-problemset` will load `lm-math` and display the symbols seen in table 2, which look significantly better with Courier than STIX' symbols.

Table 2: Shortcuts provided by the `pseudocode` environment

Input	Display	Codepoint
`<-`	←	U+2190
`->`	→	U+2192
`(/)`	∅	U+2205
`inf`	∞	U+221E
`!=`	≠	U+2260
`>=`	≥	U+2265
`<=`	≤	U+2264

To-do: Improve the font selection mechanism; maybe provide a command for each symbol?

[1] Designed for COSI 21a as taught by Dr. Antonella DiLillo

Note: If your TEX engine doesn't support UTF-8 input, the shortcuts might appear totally blank or garbled. Good luck! It will surely work with XELATEX or LuaTEX.

```
% the optional [Bar] makes [Bar] bold like the other keywords
\begin{pseudocode}[Bar]
Bar(a, n)
    Input:  two integers, a and n
    Output: a^n
    k <- n # k is a counter
    b <- 1
    c <- a
    while k > 0 do
        if k mod 2 = 0 then
            k <- k/2
            c <- c * c
        else
            k <- k - 1
            b <- b * c
    return b
\end{pseudocode}
```

`assembly` [⟨*extra options*⟩]

typesets assembly code.[2] Several considerations are taken into account; most notably, line numbers are printed as `x + n`, where n starts at 0 and counts by 4; the line number actually indicates the instruction's location in memory as an offset from the program start. Additionally, all valid instructions are treated as keywords and styled appropriately.

Any extra options are passed directly to the `listings` package.

```
\begin{assembly}
        LOAD  R4, $200       ; sum addr
        LOAD  R1, =0         ; sum
        LOAD  R2, =0         ; i
        LOAD  R3, =0         ; j
        BR    OUTER          ; we know i < 10
INNER:  ADD   R1, R3         ; sum += j
        INC   R3             ; j++
OUTER:  BLT   R3, R2, INNER  ; while j < i goto inner
        INC   R2             ; i++
        LOAD  R3, =0         ; j = 0
        BLT   R2, =10, OUTER ; while i < 10
        STORE R1, @R4        ; store sum into sum address
        HALT
\end{assembly}
```

[2] Designed for COSI 131a as taught by Dr. Liuba Shrira

`java` `[`⟨*extra options*⟩`]`
Tragically-common shorthand environment for a listing of Java code.

Any extra options are passed directly to the `listings` package.

`ganttschedule` `{`⟨*total cell count*⟩`}[`⟨*title*⟩`]`
An environment for drawing Gantt charts indicating process scheduling. The mandatory argument indicates how small the grid should be; `19` subdivides the line into 19 cells.

Within a `ganttschedule`, use the `\burst` command to indicate an active process (i.e. a process burst).

`\burst` `{`⟨*pid*⟩`}{`⟨*burst length*⟩`}`
Draw a burst for process `{`⟨*pid*⟩`}` of time length ⟨*burst length*⟩.

Note: These aren't really Gantt charts, but that's what Dr. Shrira calls them, so that's what they're called here.

Note: Because `ganttschedule` relies on `tikz`, `fp`, and `calc`, it can add significantly to document compile times. If you intend to use the `ganttschedule` environment, make sure to use the `gantt` class option or set `gantt` in `\problemsetsetup`.

```
\begin{ganttschedule}{19}
  \burst{2}{1}
  \burst{4}{1}
  \burst{3}{2}
  \burst{5}{5}
  \burst{1}{10}
\end{ganttschedule}
```

3.1 General formatting commands

`\ac` `{`⟨*acronym*⟩`}`
Typesets an acronym. The ⟨*acronym*⟩ should be lowercase (e.g. `\ac{cpu}` rather than `\ac{CPU}`). Currently, `\ac` simply delegates to `\textsc`. In the future, I'd like to support a bit of letterspacing; "for abbreviations and acronyms in the midst of normal text, use spaced small caps."[3]

`\Sc` `{`⟨*text*⟩`}`
An abbreviation for `\textsc`

`\Rm` `{`⟨*text*⟩`}`
An abbreviation for `\textrm`

`\Up` `{`⟨*text*⟩`}`
An abbreviation for `\textup`

[3] *The Elements of Typographic Style* by Robert Bringhurst, 2nd. ed, § 3.2.2

`\Bf` {⟨*text*⟩}
An abbreviation for `\textbf`

`\It` {⟨*text*⟩}
An abbreviation for `\textit`

`\Tt` {⟨*text*⟩}
An abbreviation for `\texttt`

4 Example

A brief example usage of `brandeis-problemset` follows. For a longer, more in-depth example, see `example.tex` in the `brandeis-problemset` repository.

```
\documentclass[gantt]{brandeis-problemset}
\author{Rebecca Turner}
\problemsetsetup{
  coursenumber=21a,
  instructor=Dr.\ Liuba Shrira,
  duedate=2018-10-20,
  number=3,
}
\newcommand{\io}{\ac{io}}
\newcommand{\cpu}{\ac{cpu}}
\begin{document}

\begin{problem}
  Write an assembly program!
\end{problem}

\begin{assembly}
        LOAD  R1, $200      ; A = (program location) + 200
        LOAD  R2, =1        ; i = 1
\end{assembly}

\begin{problem}
  What does this algorithm do? Analyze its worst-case running time and
  express it using big-O notation.

\begin{pseudocode}[Foo]
Foo(a, n)
  Input:  two integers, a and n
  Output: a^n
  k <- 0
  b <- 1
  while k < n do
    k <- k + 1
    b <- b * a
  return b
```

```
\end{pseudocode}
\end{problem}

$\Rm{Foo}(a, n)$ computes $a^n$, and will run in $O(n)$ time always.

\begin{problem}[number=5.4]
  Consider the following set of processes, with the length of the
  \cpu\ burst given in milliseconds:

  \begin{center}
    \begin{tabu} to 0.25\linewidth{X[1,$]rr}
      \Th{Process} & \Th{Burst time} & \Th{Priority} \\
      P_1 & 10 & 3 \\
      P_2 & 1 & 1 \\
      P_3 & 2 & 3 \\
      P_4 & 1 & 4 \\
      P_5 & 5 & 2 \\
    \end{tabu}
  \end{center}%$

  Draw a Gantt chart to illustrate the execution of these processes
  using the \ac{sjf} scheduling algorith.
\end{problem}

\begin{ganttschedule}{19}
  \burst{2}{1}
  \burst{4}{1}
  \burst{3}{2}
  \burst{5}{5}
  \burst{1}{10}
\end{ganttschedule}
\end{document}
```