The `brandeis-problemset` Document Class

Rebecca Turner*

2019/01/20

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

Class options are limited to configuration options which require the loading of fonts or other packages; "string" settings like the assignment's due date are configured either with the `\problemsetsetup` command or the commands described in section 2.2.1.

`gantt`

Load packages for the `ganttschedule` environment.

`scheme`

Define the `scheme` language for the `listings` package as well as the `scheme` shorthand environment.

`antonella`

Use Dr. Antonella DiLillio's preferred styles (Courier for code)

solutions

Include `solution` environments in compiled document.

2.2 Setting options after loading `brandeis-problemset`

`\problemsetsetup{`⟨*options*⟩`}`

Sets global `brandeis-problemset` options.

course=⟨*course name*⟩

Course name in full.

coursenumber=⟨*course number*⟩

Course name shorthand; use `21a` for "COSI 21a".

assignment=⟨*assignment name*⟩

Assignment name in full.

number=⟨*problem set number*⟩

Assignment name shorthand; use `3` for "Problem Set 3".

duedate=⟨*due date*⟩

Due date, e.g. `2018-10-18`; not parsed at all, but ISO 8601 dates are highly recommended.

instructor=⟨*course instructor*⟩

Course instructor.

author=⟨*your name*⟩

Alternate interface for the `\author` command.

date=⟨*document date*⟩

Alternate interface for the `\date` command.

codefont=⟨*fontspec font name*⟩

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=⟨`true`|`false`⟩ `false`

Load packages for the `ganttschedule` environment.

antonella=⟨`true`|`false`⟩ `false`

Use Dr. Antonella DiLillio's preferred styles (Courier for code)

solutions=⟨`true`|`false`⟩ `false`

Include `solution` environments in compiled document.

2.2.1 Configuration commands

`brandeis-problemset` additionally provides a number of configuration commands for setting a single opption 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*⟩`]{`⟨*fontspec 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 customized document class for each course. There's no reason to install these to some system-wide directory; it makes sense for them to live in the same directory as the problem set source files. For instance, `cosi21a.cls` might read:

```
\LoadClass[antonella]{brandeis-problemset}

% 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{cosi21a}
% stuff specific to this assignment
\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.

`\begin{problem}[`⟨*options*⟩`]...\end{problem}`

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. ⟨*options*⟩ may include:

`title`=⟨*problem title*⟩

Displayed after "Problem" and the problem's number.

`number`=⟨*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`⟩ `true`

Add a pagebreak before the problem?

`label`=⟨*problem label*⟩

Adds a custom label to the problem with `\label` that can be used with `\ref`. I recommend prefixing your problem labels with `p:` as in `p:big-o-proofs`.

`part`=⟨*part name*⟩

Indicates that this problem starts a new "part" of the assignment; actually calls `\part` under the hood.

partlabel=⟨*part label*⟩

Adds a custom label to this part in the same fashion as the `label` key.

Vertical material is allowed in a `problem`.

`\begin{solution}...\end{solution}`

Defines a solution for a problem; a solution prints in blue and is excluded from the compiled document entirely unless the `solutions` package option is given.

In this way, the same `.tex` file can serve as both a postable assignment prompt and an answer key.

NOTE The style of solutions is customizable by redefining `\solutionstyle`; it's defined to `\color{blue}` by default.

`\subproblem[`⟨*problem description*⟩`]`

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

`\Th[`⟨*column spec*⟩`]{`⟨*header text*⟩`}`

Typesets a table header in bold-face. ⟨*column spec*⟩ defaults to `l`. Useful for when a column is wrapped in a math environment.

`\begin{pseudocode}[`⟨*keywords*⟩`]...\end{pseudocode}`

Prints pseudocode.[1]

Several "shortcuts," which replace a source-code sequence like `->` with a symbol like →, are shown in table 1.

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 1, which look significantly better with Courier than `stix`' symbols.

Table 1: 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

TO-DO 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}
```

`\begin{assembly}[`⟨*listings options*⟩`]...\end{assembly}`

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.

⟨*listings options*⟩ is 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

`\begin{java}[`⟨*listings options*⟩`]...\end{java}`

Tragically-common shorthand environment for a listing of Java code.
⟨*listings options*⟩ is passed directly to the `listings` package.

`\begin{scheme}[`⟨*listings options*⟩`]...\end{scheme}`

Shorthand environment for a listing of Scheme code, useful for COSI 121b. Requires the `scheme` package option to be loaded.
⟨*listings options*⟩ is passed directly to the `listings` package.

`\begin{ganttschedule}[`⟨*total cell count*⟩`]...\end{ganttschedule}`

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.
To use the `ganttschedule` environment, make sure to use the `gantt` package option.
Within a `ganttschedule`, use the `\burst` command to indicate an active process (i.e. a process burst).

NOTE The charts `ganttschedule` draws aren't actually really proper Gantt charts, which can indicate parallel activities; however, that's what Liuba calls them, so that's what they're called here.

`\pid{`⟨*pid*⟩`}{`⟨*burst length*⟩`}`

Draw a burst for process ⟨*pid*⟩ of time length ⟨*burst length*⟩.

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

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`. If you fail to include the `gantt` option, you will see an error message:

```
! Package brandeis-problemset Error: ganttschedule enviornment
   not loaded in preamble.

See the brandeis-problemset package documentation for
   explanation.
Type  H <return>  for immediate help.
l.4 \burst
    {1}{1}
? H
Did you mean to use the 'gantt' option for the
   brandeis-problemset document class?
```

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`.

`\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}}
```

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

```
\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}
```

5 Changelog

0.4.3 Rebecca Turner (2019-01-20) — Fixed license typos, corrected documentation PDF.

0.4.2 Rebecca Turner (2019-01-19)

Added

- `author` and `date` keys added to `\problemsetsetup` to simplify class-wide configuration.

Fixed

- Fixed definitions for `\duedate`, `\instructor`, etc. to avoid spurious errors due to undefined commands.

Changed

- Translated documentation to the new `ltxguidex` document class for added beauty.
- Re-licensed `brandeis-problemset` to the LPPL v1.3c for easy transfer of maintenence in the future.

0.4.1 Rebecca Turner (2019-01-03) — Updated `scheme` environment to properly recognize all primitive functions, added syntax coloring to all code.

0.4.0 Rebecca Turner (2018-12-20)

Added

- `solution` environment and `solutions` class option.
- `scheme` shorthand environment and `scheme` class option.

Fixed

- Boolean class options being overwritten by keys defined for `\problemsetsetup`.
- Title-formatting errors

Removed

- Assignment- and course-specific class options `duedate`, `assignment`, `instructor`, and `course`. These settings should be configured with either `\problemsetsetup` or their specific commands. (`\duedate`, `\instructor`, etc.).

0.3.0 Rebecca Turner (2018-10-24)

Added

- This changelog.
- Support for `\parts` and referencing problems.
- Options to `problem` environment: `part`, `label`, and `partlabel`.
- `\maketitle` (contrast with `\maketitlepage`).

0.2.0 Rebecca Turner (2018-10-20)

Changed

- Class renamed to from `problemset` to `brandeis-problemset`.

Added

- A license header.
- `ganttschedule` environment.
- Additional keywords for `pseudocode` environment: `and`, `or`, `nil`, and `len`.
- `\ac` command for acronyms.
- An example document.

0.1.0 Rebecca Turner (2018-10-19) — Initial beta as `problemset`.