

The `ran_toks` Package

D. P. Story

Email: `dpstory@uakron.edu`

processed December 30, 2019

Contents

1 Alternate package name: <code>ran-toks</code>	3
2 Commands for controlling the process	3
3 Utility commands	4
4 The main commands	7
4.1 Additional user access commands	9
5 Commands that support a DB application	10
6 Index	12
7 Change History	14
1 <code>(*package)</code>	

Description. This short package randomizes a list of tokens. The command, `\ranToks`, takes one argument, which is a list of tokens:

```
\ranToks{\langle name\rangle}
{
    {\langle tok_1\rangle}{\langle tok_2\rangle}...{\langle tok_n\rangle}
```

The command defines a series of n internal commands, one for each of the tokens. The definitions are essentially randomized. The randomized tokens are accessed through the command `\useRanTok`. For example

```
\useRanTok{1}, \useRanTok{2},..., \useRanTok{n}
```

gives a random listing of the n tokens. These can be arranged on the page as desired.

There is a second construct, designed for more elaborate randomization.

```

\bRTVToks{\name}
\begin{rtVW}
    (some content)
\end{rtVW}
...
...
\begin{rtVW}
    (some content)
\end{rtVW}
\end{bRTVToks}

```

The contents of each of the `rtVW` environments are written to the computers hard drive, then input back in random order, using `\useRanTok`, eg,

```
\useRanTok{1}, \useRanTok{2}, ..., \useRanTok{n}
```

Other details are left to the readers' imagination.

Requirements. As of this writing, we require only the `verbatim` package and `random.tex`, the package was written by Donald Arseneau.

```
2 \RequirePackage{verbatim}
```

Input random.tex. Input `random.tex` if not already input.

```
3 \@ifundefined{nextrandom}{\input{random.tex}}{}
```

We redefine `\nextrandom` from `random.tex` to save the initializing seed.

```

4 \def\nextrandom{\begingroup
5   \ifnum\randomi<\@ne % then initialize with time
6     \global\randomi\time
7     \global\multiply\randomi388 \global\advance\randomi\year
8     \global\multiply\randomi31 \global\advance\randomi\day
9     \global\multiply\randomi97 \global\advance\randomi\month
10    \message{Randomizer initialized to \the\randomi.}%
11    \nextrandom \nextrandom \nextrandom

```

Save the initial seed value to `\rtInitSeedValue`.

```

12   \xdef\InitSeedValue{\the\randomi}%
13 \fi
14 \count@ii\randomi
15 \divide\count@ii 127773 % modulus = multiplier * 127773 + 2836
16 \count@\count@ii
17 \multiply\count@ii 127773
18 \global\advance\randomi-\count@ii % random mod 127773
19 \global\multiply\randomi 16807
20 \multiply\count@ 2836
21 \global\advance\randomi-\count@
22 \ifnum\randomi<\z@ \global\advance\randomi 2147483647\relax\fi
23 \endgroup
24 }
```

The code for this package was taken from the `dps` package, and modified suitably. We use several token registers and count registers. This can probably be optimized.

```

25 \newtoks\rt@listIn \rt@listIn={}
26 \newtoks\rt@ newListIn \rt@ newListIn={}
27 \newtoks\rt@listOut \rt@listOut={}
28 \newcount\rt@nMax
29 \newcount\rt@nCnt
30 \newcount\rt@getRanNum
31 \newif\ifrtdebug \rtdebugfalse
32 \newif\ifwerandomize \werandomizetrue
33 \newif\ifsaveseed\saveseedtrue
34 \newwrite\rt@Verb@write
35 \def\rt@nameedef#1{\expandafter\edef\csname #1\endcsname}
36 </package>
37 <*altpkgname>
```

1 Alternate package name: `ran-toks`

CTAN lists this package (`ran_toks`) as `ran-toks`, so we'll create a dummy package by that name.

```

38 \NeedsTeXFormat{LaTeX2e}
39 \ProvidesPackage{ran-toks}
40 [2019/12/28 v1.0 ran-toks Alt-name (dps)]
41 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{ran_toks}}
42 \ProcessOptions
43 \RequirePackage{ran_toks}[2019/12/28]
44 </altpkgname>
45 <*package>
```

2 Commands for controlling the process

<code>\ranToksOn</code>	These two turn on and turn off randomization.
<code>\ranToksOff</code>	<code>46 \def\ranToksOn{\werandomizetrue}</code> <code>47 \def\ranToksOff{\werandomizefalse}</code>
<code>\useThisSeed</code>	initializes the random number generator. Use this to reproduce the same sequence of pseudo-random numbers from an earlier run. We also set <code>\saveseedfalse</code> so we do not write the initial seed to the disk. <code>48 \def\useThisSeed#1{\saveseedfalse\randomi=#1}</code> <code>49 \onlypreamble\useThisSeed</code>
<code>\useLastAsSeed</code>	initializes the random number generator using the last random seed. If the file <code>\jobname_rt.sav</code> does not exist, the generator will be initialized using time and date data. <code>50 \def\useLastAsSeed{\rt@useLastAsSeed}</code> <code>51 \onlypreamble\useLastAsSeed</code>

```

52 \def\rt@useLastAsSeed{%
53   \IfFileExists{\jobname_rt.sav}{%
54     \PackageInfo{ran_toks}{Inputting \jobname_rt.sav}%
55     \CIfundefined{readsavfile}{\newread\readsavfile}{}{%
56       \openin\readsavfile=\jobname_rt.sav
57       \read\readsavfile to \InitSeedValue
58       \read\readsavfile to \lastRandomNum
59       \closein\readsavfile
60       \randomi=\lastRandomNum

```

When \useLastAsSeed, the last becomes the first.

```

61   \xdef\InitSeedValue{\the\randomi}
62 }{%
63   \PackageInfo{ran_toks}{\jobname_rt.sav cannot
64     be found, \MessageBreak
65     using the random initializer}%
66 }%
67 }
68 \CIfundefined{aeb@randomizeChoices}{%
69   \let\inputRandomSeed\useLastAsSeed
70   \let\useRandomSeed\useThisSeed}{}%

```

3 Utility commands

A standard \verbatim write used in exerquiz and other package in the AeB family.

```

71 \def\verbatimwrite{@bsphack
72   \let\do\@makeother\dospecials
73   \catcode`\^^M\active \catcode`\^^I=12
74   \def\verbatim@processline{%
75     \immediate\write\verbatim@out
76     {\the\verbatim@line}}%
77   \verbatim@start}
78 \def\endverbatimwrite{@esphack}
79 \def\rt@IWW0{\immediate\write\verbatim@out}

```

We write only if \ifsaveseed is true.

```

80 \def\InitSeedValue{\the\randomi}
81 \def\rt@writeSeedData{\ifsaveseed
82   \CIfundefined{saveinfo}{\newwrite\saveinfo}{}{%
83     \immediate\openout\saveinfo \jobname_rt.sav
84     \let\verbatim@out\saveinfo
85     \def\rt@msgii{initializing seed value}%
86     \def\rt@msgii{last random number used}%
87     \uccode`c='\\uppercase{%
88     \rt@IWW0{\InitSeedValue\space c \rt@msgii}%
89     \rt@IWW0{\the\randomi\space c \rt@msgii}}\immediate
90     \closeout\saveinfo\fi}

```

Save the initial seed value to hard drive.

```

91 \AtEndDocument{\rt@writeSeedData}%

```

`\rt@populateList{<n>}` is a utility command, its argument $<n>$ is a positive integer, and it generates a list of the form $\{\{1\}\,\{2\}\dots\,\{n\}$ and is held in the token register `\rt@listIn`. This listing is later randomly permuted by `\rt@RandomizeList`.

```
92 \def\rt@populateList#1{\rt@listIn={} \rt@nCnt\z@
93   \@whilenum\rt@nCnt<#1\do{\advance\rt@nCnt\@ne
94     \edef\rt@listInHold{\the\rt@listIn\noexpand\{\the\rt@nCnt\}%
95     \rt@listIn=\expandafter{\rt@listInHold}}}
```

`\rt@RandomizeList {<n>}` is the command that gets the process of randomizing the input list going. The argument is the number $<n>$ of tokens. If `\werandomize` is false, it just returns the input list; otherwise, it calls `\rt@randomizeList` to actually do the work.

```
96 \def\rt@RandomizeList#1{\global
97   \rt@listIn={} \global\rt@ newListIn={} \global\rt@listOut={}
98   \rt@nMax=#1\relax\rt@populateList{\the\rt@nMax}%
99   \ifwerandomize
100     \expandafter\rt@randomizeList\else
101     \global\rt@listOut=\expandafter{\the\rt@listIn}\fi}
```

Save the list out as `\rt@BaseName-List` for later retrieval. This is the randomized list of integers for this base name.

```
102 \global\rt@nameedef{\rt@BaseName-List}{\the\rt@listOut}}
```

`\rt@randomizeList` randomizes the list of consecutive integers, and leaves the results,

```
\{\{k_1\}\,\{k_2\}\dots\,\{k_n\}
```

in the token register `\rt@listOut`. `\rt@randomizeList` is a loop, looping between itself and `\rt@loopTest`.

```
103 \def\rt@randomizeList{\let\\=\rt@processi
104   \setrannum{\rt@getRanNum}{1}{\the\rt@nMax}%
105 \ifrtdebug\typeout{\string\rt@getRanNum=\the\rt@getRanNum}\fi
106   \rt@nCnt\z@
107 \ifrtdebug\typeout{LISTING: \the\rt@listIn}\fi
108   \the\rt@listIn
109   \rt@loopTest
110 }
111 \def\rt@loopTest{\advance\rt@nMax\m@ne
112   \ifnum\rt@nMax>\z@
113     \def\rt@next{%
114       \rt@listIn=\expandafter{\the\rt@newListIn}%
115       \rt@newListIn={} \rt@randomizeList}%
116     \else
117       \let\rt@next\relax
118       \global\rt@listOut=\expandafter{\the\rt@listOut}%
119       \ifrtdebug
120         \typeout{Final Result: \string\rt@listOut=\the\rt@listOut}\fi
121     \fi\rt@next
122 }
```

In `\rt@randomizeList`, we `\let\\=\rt@processi` before dumping the contents of `\rt@listIn`. We then go into a loop `\rt@loopTest`. `\rt@getRanNum` is the random integer between 1 and `\rt@nMax`.

```

123 \def\rt@processi#1{\advance\rt@nCnt\@ne
124   \ifnum\rt@nCnt=\rt@getRanNum
125     \edef\rt@listOutHold{\the\rt@listOut}%
126     \global\rt@listOut=\expandafter{\rt@listOutHold\\{#1}}%
127     \ifrtdebug\typeout{Found it: \string\\{#1}}%
128     \typeout{New \string\rt@listOut: \the\rt@listOut}\fi
129   \else
130     \edef\rt@listInHold{\the\rt@newListIn}%
131     \rt@newListIn=\expandafter{\rt@listInHold\\{#1}}%
132     \ifrtdebug\typeout{\string\rt@newListIn: \the\rt@newListIn}\fi
133   \fi
134 }

We perform modular arithmetic when the index of \useRanTok is too large.
\rt@modarith performs modular arithmetic on its arguments (#1 mod #2) and
returns the result in the macro \rt@mod.

135 \def\rt@modarith#1#2{\count\z@=#1\relax\count\tw@=#1\relax
This macro uses \dimen0 and \dimen2, so it should be called within a group.
136   \advance\count\z@\m@ne\divide\count\z@ #2\relax
137   \multiply\count\z@ #2\relax
138   \advance\count\tw@-\count\z@
139   \edef\rt@mod{\the\count\tw@}

\rt@badIndex  Warning messages, these are \rt@badIndex and \rt@badTokName.
\rt@badTokName 140 \def\rt@badIndex#1#2{\PackageWarningNoLine{ran_toks}
141   {The argument of \string\useRanTok{#1} on line
142    \the\inputlineno\space is\MessageBreak
143    greater than \string\nToksFor{#2} (\nToksFor{#2}),
144    instead will use\MessageBreak
145    \string\useRanTok{\rt@mod}, obtained from modular
146    arithmetic.\MessageBreak
147    You might want to fix this}
148 }
149 \def\rt@badTokName#1{%
150   \PackageWarningNoLine{ran_toks}
151   {The token list '#1' on line \the\inputlineno\space
152    is undefined,\MessageBreak
153    possibly simply misspelled; check spelling.\MessageBreak
154    If undefined, use \string\ranToks\space or \string\bRTVToks/%
155    \string\@RTVToks\space\MessageBreak
156    to define a list with the name '#1'}%
157 }
158 \def\rt@warnTokName#1{%
159   \PackageWarningNoLine{ran_toks}
160   {The token list '#1' on line \the\inputlineno\space
161    is already defined,\MessageBreak
162    will overwrite this list}%
163 }

```

4 The main commands

`\ranToks{<token-list>}` takes one argument, `{<token-list>}`, a list of tokens. It randomizes them. The randomized listing can be accessed using `\useRanTok`.

```

164 \def\ranToks#1{\begingroup
165   \useRTName{#1}%
166   \r@nToks
167 }
168 \long\def\r@nToks#1{\rt@nMax\z@\r@ndToks#1\rt@NIL}
169 \def\rt@NIL{@nil}

```

`\useRTName{<name>}` sets the base name (use prior to the use of `\useRanTok`).

```

170 \newcommand{\useRTName}[1]{\gdef\rt@BaseName{#1}}%
171 \let\rt@BaseName\empty

```

`\bRTVToks{<name>}` `\bRTVToks` and `\eRTVToks` enclose a series of `rtVW` environments. The single argument is the name of this set of verbatim write “tokens”.

```
172 \newcommand{\bRTVToks}[1]{\rt@nCnt\z@\useRTName{#1}}
```

`\eRTVToks` At the end of the `rtVW` environments, initiated by `\bRTVToks`, the `\eRTVToks` command saves the number of tokens counted, and randomizes the access to the contents of the `rtVW` environments, this done by `\r@nVToks`.

```

173 \newcommand{\eRTVToks}{\global
174   \rt@nameedef{\rt@BaseName Cnt}{\the\rt@nCnt}%
175   \expandafter\r@nVToks\expandafter{\rt@BaseName}}

```

`rtVW` `\rtVW` is a verbatim write environment. It saves its contents to the file `\jobname_\rt@BaseName\the\rt@nCnt.cut`. The file is later input back into the source file in a random way.

```
176 \def\reVerbEnd{\ifhmode\unskip\fi}
```

Insert the hook `\rtVWHook` prior to writing the verbatim content. The default is `\relax`.

```

177 \def\rtVWHook#1{\def\@rgi{#1}\ifx\@rgi\empty
178   \let\RTVWHook\relax\else\def\RTVWHook{#1}\fi}
179 \rtVWHook{}
180 \newenvironment{rtVW}{\global\advance\rt@nCnt\@ne
181   \immediate\openout\rt@Verb@write
182   \jobname_\rt@BaseName\the\rt@nCnt.cut
183   \let\verbatim@out\rt@Verb@write
184   \rt@IWVO{\string\RTVWHook}%
185   \verbatim@write
186 }{%
187   \endverbatim@write
188   \immediate\write\rt@Verb@write{\string\reVerbEnd}%
189   \immediate\closeout\rt@Verb@write
190 }

```

`\r@nVToks` randomizes the contents of the `rtVW` environment.

```

191 \def\r@nVToks#1{\begingroup
192   \gdef\rt@BaseName{#1}%
193   \expandafter\rt@nMax\@nameuse{#1Cnt}%
194   \rt@listIn={} \rt@nCnt=0\relax\let\rt@listInHold\@empty
195   \@whilenum\rt@nCnt<\rt@nMax\do{\advance\rt@nCnt\@ne
196     \edef\rt@listInHold{%
197       \the\rt@listIn\noexpand\rt@inputVerb{#1\the\rt@nCnt}}%
198     \rt@listIn=\expandafter{\rt@listInHold}\ifrtdebug
199     \typeout{\string\r@nVToks: \the\rt@listIn}\fi
200     \expandafter\rt@nToks\expandafter{\the\rt@listIn}%
201 \def\rt@inputVerb#1{\input{\jobname_\#1.cut}}

```

`\r@ndToks` is main looping command for `\ranToks` and `\eRTVToks` (through `\r@nVToks`). If the ending token `\rt@NIL` is detected, we break off and go to `\rt@endToks`.

```

202 \def\rt@PAR{\par}
203 \long\def\r@ndToks#1{\def\rt@rgi{#1}%
  If the current argument is \par, we skip it
204   \ifx\rt@rgi\rt@PAR\def\rt@next{\r@ndToks}\else
205     \advance\rt@nMax\@ne
206     \global\@namedef{rtTok\the\rt@nMax\rt@BaseName}{#1}%
207     \def\rt@next{\@ifnextchar\rt@NIL
208       {\rt@endToks\@gobble}{\r@ndToks}}\fi\rt@next}

```

`\rt@performRanDefns{<n>}` The `\rt@performRanDefns` performs code that is repeated in several other macros: `\rt@endToks`, `\reorderRanToks`, and `\copyRanToks`. It randomizes the list `\rt@RandomizeList`, then assignments the randomized list to the definitions.

```

209 \def\rt@performRanDefns#1{%
  Now we randomize the order of the integers 1, 2,...#1.
210   \rt@RandomizeList{#1}\rt@nCnt\z@
  Now we randomize the definitions. We \let\\=\rt@ssign, then let loose the tokens!
211   \let\\=\rt@ssign\the\rt@listOut}

```

`\rt@endToks` The final destination for `\r@ndToks`.

```

212 \def\rt@endToks{\global
  Save the number of tokens counted
213   \rt@nameedef{nMax4\rt@BaseName}{\the\rt@nMax}%
214   \rt@performRanDefns{\the\rt@nMax}\endgroup}

```

`\reorderRanToks{<name>}` The `\reorderRanToks` command reorders (or re-indexes) the family with name `<name>` (#1).

```

215 \def\reorderRanToks#1{\begingroup\useRTName{#1}\expandafter
216   \ifx\csname nMax4#1\endcsname\relax
    Document author has not run \ranToks yet for this basename (#1)
217   \rt@badTokName{#1}\else

```

Good to go. We reorder this list.

```
218     \rt@performRanDefns{\@nameuse{nMax4#1}}\fi  
219 \endgroup}
```

\copyRanToks{*name1*}{*name2*} Use this command to copy *name1* to *name2*. This gives a randomization of the same list, without affecting the original order of *name1*.

```
220 \newcommand\copyRanToks[2]{\begingroup\expandafter  
221   \ifx\csname nMax4#1\endcsname\relax
```

Source list is not defined

```
222     \rt@badTokName{#1}%
223   \else\expandafter
```

Source list is defined

```
224   \ifx\csname nMax4#2\endcsname\relax
```

Destination list is not defined, which is good in this instance. This is the case we copy the list.

```
225     \useRTName{#2}\global  
226     \rt@nameedef{nMax4#2}{\@nameuse{nMax4#1}}%
227     \rt@nCnt=\csname nMax4#2\endcsname\relax  
228     \@whilenum\rt@nCnt>\z@\do{\global  
229       \rt@nameedef{rtTok\the\rt@nCnt#2}%
230       {\noexpand\@nameuse{rtTok\the\rt@nCnt#1}}%
231       \advance\rt@nCnt\m@ne}%
232     \rt@performRanDefns{\@nameuse{nMax4#2}}\else
```

Destination list is defined already, warn the user.

```
233   \rt@warnTokName{#2}\fi  
234 \fi  
235 \endgroup}
```

\rt@ssign{*name*} makes the assignments that are expanded by \useRanTok. We \let the assignment \let\\=\rt@ssign in \rt@endToks, just before we dump out the contents of \the\rt@listOut.

```
236 \def\rt@ssign#1{\advance\rt@nCnt\@ne\global  
237   \rt@nameedef{rtRanTok\the\rt@nCnt\rt@BaseName}{\noexpand  
238     \@nameuse{rtTok#1\rt@BaseName}}}
```

4.1 Additional user access commands

\nToksFor{*name*} expands the the number of tokens whose name is *name* (#1).

```
239 \newcommand\nToksFor[1]{\expandafter  
240   \ifx\csname nMax4#1\endcsname\relax  
241     \textbf{??}\rt@badTokName{#1}\else  
242     \@nameuse{nMax4#1}\fi}
```

\rtTokByNum[*name*]{*num*} is an internal macro, but it can be used publicly. The argument of it is an integer, eg, \rtTokByNum{3} is the third token, as listed in the order given in the argument of \ranToks.

```

243 \newcommand{\rtTokByNum}[2][\rt@BaseName]{\expandafter
244   \ifx\csname nMax4#1\endcsname\relax
245     \textbf{??}\rt@badTokName{#1}\else
246     \nameuse{rtTok#2#1}\expandafter\ignorespaces\fi}
\useRanTok[⟨name⟩]{⟨num⟩} After \ranToks has been executed, the user has access to the randomized tokens through \useRanTok. The argument ⟨num⟩ is an integer 1 through max.
247 \newcommand{\useRanTok}[2][\rt@BaseName]{\bgroup
248   \expandafter\ifx\csname nMax4#1\endcsname\relax
249     \rt@badTokName{#1}\else
250     \ifnum#2>\nToksFor{#1}\rt@modarith{#2}{\nToksFor{#1}}%
If index (#2) is greater than array length, use modular arithmetic to resolve the issue, and send a warning to the user.
251   \rt@badIndex{#2}{#1}\nameuse{rtRanTok\rt@mod#1}\else
252     \nameuse{rtRanTok#2#1}\fi\fi\egroup}
\displayListRandomly[⟨prior⟩][⟨post⟩]{⟨name⟩} lists all items in the list as passed by the required argument. For expanding in a list environment, use \item as the optional argument. Designed for listing all question in an eqexam document in random order.
253 \newcommand{\displayListRandomly}[1][]{\bgroup\def\rt@prior{#1}%
254   \displ@yListRandomly}
255 \newcommand{\displ@yListRandomly}[2][]{\rt@nCnt\z@%
256   \expandafter\ifx\csname nMax4#2\endcsname\relax
257     \rt@rgi\space\textbf{??}\rt@badTokName{#2}#1%
258   \else
\i Within the optional arguments, we define \i, \first, \last, and \lessone to
\first do some logic on the arguments. These four macro are defined locally and not
\last available outside the command \displayListRandomly.
\lessone 259   \def\rt@post{#1}\useRTName{#2}\let\i\rt@nCnt
260   \def\first{1}\edef\last{\nameuse{nMax4#2}}%
261   \tempcnta\last \advance\tempcnta\m@ne
262   \edef\lessone{\the\tempcnta}%
263   \whilenum\rt@nCnt<\last\advance\rt@nCnt\m@ne
264   \do{\rt@prior\useRanTok{\the\rt@nCnt}\rt@post}%
265 \fi
266 \egroup}

```

5 Commands that support a DB application

We begin with some utility commands to help parse the argument of \useProbDBs.

```

267 \def\rt@gettonil#1\@nil{\def\to@nilarg{#1}}
268 \def\rt@ifspc{\ifx\@let@token\@sptoken
269   \let\rt@next\rt@xifspc\else
270   \let\rt@next\rt@gettonil\fi\rt@next}
271 \begingroup
272 \def\@:{\rt@xifspc}

```

```

273 \expandafter\gdef\:\ {\futurelet\@let@token\rt@ifspc}
274 \endgroup
275 \def\rt@strpspc{\futurelet\@let@token\rt@ifspc}

```

\useTheseDBs{*list*} Inputs any files included in the comma-delimited list. The base names need only be listed, as the extension is assumed to be .tex. The command \useProbDBs can only be used in the preamble. Refer to the demo file `mc_db.tex` for an illustration of its intended use.

```

276 \def\ProbDBWarningMsg#1{\filename@parse{#1}
277   \PackageWarning{ran_toks}
278   {The file \filename@area\filename@base.\ifx\filename@ext\relax
279    tex\else\filename@ext\fi\space cannot be found}}
280 \def\useTheseDBs#1{\def\rt@dblist{#1}\ifx\rt@dblist\empty\else
281   \let\rt@DB@List\empty
282   \edef\temp@expand{\noexpand\for\noexpand\@@tmp:=\rt@dblist}%
283   \temp@expand\do{\ifx\@@tmp\empty\else
284     \expandafter\rt@strpspc\@@tmp\@nil\edef\@@tmp{\to@nilarg}%
285     \edef\rt@nextDB{\noexpand
286       \InputIfFileExists{\@@tmp}{}\noexpand
287       \ProbDBWarningMsg{\@@tmp}}\%
288   \toks\tw@=\expandafter{\rt@DB@List}\%
289   \toks@=\expandafter{\rt@nextDB}\%
290   \edef\rt@DB@List{\the\toks\tw@\space\the\toks@}\fi
291 } \expandafter\rt@DB@List\fi}
292 \let\useProbDBs\useTheseDBs

```

```
293 </package>
```

6 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	I
\%	87
\:	272, 273
\@bsphack	71
\@let@token	268, 273, 275
\@makeother	72
\@onlypreamble	49, 51
\@rgi	177
\@sptoken	268
\^	73
A	
\active	73
\AtEndDocument	91
B	
\bRTVToks	154, <u>172</u>
C	
\copyRanToks	<u>220</u>
\CurrentOption	41
D	
\day	8
\DeclareOption	41
\displ@yListRandomly	254, 255
\displayListRandomly	10, 253
\divide	15, 136
\dospecials	72
E	
\egroup	252, 266
\endverbatimwrite	78, 187
environments:	
rtVW	<u>176</u>
\eRTVToks	155, <u>173</u>
F	
\filename@area	278
\filename@base	278
\filename@ext	278, 279
\filename@parse	276
\first	10, 260
\futurelet	273, 275
I	
\i	10
\IfFileExists	53
\ifhmode	176
\ifrtdebug	31, 105, 107, 119, 127, 132, 198
\ifsaveseed	33, 81
\ifwerandomize	32, 99
\InitSeedValue	12, 57, 61, 80, 88
\input	3, 201
\InputIfFileExists	286
\inputlineno	142, 151, 160
\inputRandomSeed	69
L	
\last	10, 260, 261, 263
\lastRandomNum	58, 60
\lessone	10, 262
M	
\m@ne	111, 136, 231, 261
\month	9
\multiply	7–9, 17, 19, 20, 137
N	
\NeedsTeXFormat	38
\newwrite	34, 82
\nextrandom	4, 11
\nToksFor	9, 143, 239, 250
O	
\openin	56
\openout	83, 181
P	
\PackageInfo	54, 63
\PackageWarning	277
\PackageWarningNoLine	140, 150, 159
\PassOptionsToPackage	41
\ProbDBWarningMsg	276, 287
\ProcessOptions	42
\ProvidesPackage	39
R	
\r@ndToks	8, 168, 203, 204, 208
\r@nToks	166, 168, 200
\r@nVToks	8, 175, 191, 199

\randomi	5–10, 12, 14, 18, 19, 21, 22, 48, 60, 61, 80, 89	9, 211, 236
\ranToks	154, <u>164</u>	275, 284
\ranToksOff	3, 47	50, 52
\ranToksOn	3, 46	34, 181, 183, 188, 189
\read	57, 58	158, 233
\readsavfile	55–59	81, 91
\reorderRanToks	215	269, 272
\RequirePackage	2, 43	31
\reVerbEnd	176, 188	9, 243
\rt@badIndex	6, 140, 251	<u>176</u>
\rt@badTokName	6, 149, 217, 222, 241, 245, 249, 257	178, 184
\rt@BaseName	102, 170, 171, 174, 175, 182, 192, 206, 213, 237, 238, 243, 247	177, 179
\rt@DB@List	281, 288, 290, 291	S
\rt@dblist	280, 282	\saveseedfalse
\rt@endToks	8, 208, 212	82–84, 90
\rt@getRanNum	30, 104, 105, 124	\saveseedinfo
\rt@gettonil	267, 270	\saveseedtrue
\rt@ifspc	268, 273, 275	33
\rt@inputVerb	197, 201	\setrnum
\rt@IWVO	79, 88, 89, 184	104
\rt@listIn	25, 92, 94, 95, 97, 101, 107, 108, 114, 194, 197–200	T
\rt@listInHold	94, 95, 130, 131, 194, 196, 198	\temp@expand
\rt@listOut	27, 97, 101, 102, 118, 120, 125, 126, 128, 211	282, 283
\rt@listOutHold	125, 126	\textbf
\rt@loopTest	109, 111	241, 245, 257
\rt@mod	139, 145, 251	\time
\rt@modarith	6, 135, 250	\to@nilarg
\rt@msgi	85, 88	267, 284
\rt@msgii	86, 89	\toks
\rt@nameedef	35, 102, 174, 213, 226, 229, 237	288, 290
\rt@nCnt	29, 92–94, 106, 123, 124, 172, 174, 180, 182, 194, 195, 197, 210, 227–231, 236, 237, 255, 259, 263, 264	U
\rt@ newListIn	26, 97, 114, 115, 130–132	\uccode
\rt@next	113, 117, 121, 204, 207, 208, 269, 270	87
\rt@nextDB	285, 289	\uppercase
\rt@NIL	168, 169, 207	\useLastAsSeed
\rt@nMax	28, 98, 104, 111, 112, 168, 193, 195, 205, 206, 213, 214	3, 50, 51, 69
\rt@PAR	202, 204	\useProbDBs
\rt@performRanDefns	8, 209, 214, 218, 232	292
\rt@populateList	4, 92, 98	\useRandomSeed
\rt@post	259, 264	70
\rt@prior	253, 264	\useRanTok
\rt@processi	103, 123	10, 141, 145, 247, 264
\rt@RandomizeList	5, 96, 210	\useRTName
\rt@randomizeList	5, 100, 103, 115	165, <u>170</u> , 172, 215, 225, 259
\rt@rgi	203, 204, 257	\useTheseDBs
		\useThisSeed
		276
		3, 48, 49, 70
		V
		\verbatim@line
		76
		\verbatim@out
		75, 79, 84, 183
		\verbatim@processline
		74
		\verbatim@start
		77
		\verbatimwrite
		71, 185
		W
		\werandomizefalse
		47
		\werandomizetrue
		32, 46
		\write
		75, 79, 188
		Y
		\year
		7

7 Change History

v1.0b (2013/07/29)		Fixed a bug, when the first two tokens #1 are the same, we get an incorrect decision	8
General: Added \displayListRandomly	10		
v1.0c (2013/08/03)		v1.1 (2017/05/04)	
General: Save the initial seed value to \rtInitSeedValue.	2	General: Add modular arithmetic to resolve case where index is greater than length	10
v1.0d (2013/08/03)		Added second optional argument to \displayListRandomly	10
General: Added conditional input of random.tex .	2	Save out list for later use	5
v1.0e (2016/02/06)		v1.2 (2019/12/28)	
General: Added optional argument to \displayListRandomly	10	General: Added dummy package ran-toks	3
		\rtVW: Defined \rtVWHook	7