Ref.

Powd. Pat. Debye-Scherrer (114.6 mm; CuKα; Visual f).

<table>
<thead>
<tr>
<th>STRONGEST LINES</th>
<th>LARGEST d SPACINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.301 100</td>
<td>8.765 4</td>
</tr>
<tr>
<td>4.872 90</td>
<td>7.481 3</td>
</tr>
<tr>
<td>4.364 80</td>
<td>5.512 9</td>
</tr>
<tr>
<td>3.164 80</td>
<td>5.934 3</td>
</tr>
<tr>
<td>3.102 80</td>
<td>4.572 3</td>
</tr>
<tr>
<td>2.924 55</td>
<td>4.645 3</td>
</tr>
</tbody>
</table>

Struct. Cell. \( P1 - C_1 \); \( Z = 1 \);
\( a = 9.497(1) \), \( b = 11.443(2) \), \( c = 10.841(2) \)
\( \alpha = 120^\circ 30(1)' \), \( \beta = 92^\circ 06(1)' \), \( \gamma = 55^\circ 50(1)' \)
\( a:b:c = 0.330:1:0.947 \)

Chem. Substitution of Zn for Pb noted in some samples.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZnO</td>
<td>2.7</td>
<td>3.93</td>
</tr>
<tr>
<td>PbO</td>
<td>73.0</td>
<td>70.5</td>
</tr>
<tr>
<td>Cr₂O₃</td>
<td>19.7</td>
<td>19.5</td>
</tr>
<tr>
<td>SiO₂</td>
<td>3.9</td>
<td>3.2</td>
</tr>
<tr>
<td>F</td>
<td>1.2</td>
<td>5.1</td>
</tr>
<tr>
<td>O ≡ F</td>
<td>-0.5</td>
<td>-2.1</td>
</tr>
</tbody>
</table>

Total 100.0 100.0

1. ZnF₂[Pb₂(CrO₄)₂SiO₄]₂
2. Average of several partial analyses.

Opt. Thin section shows feeble pleochroism with \( Y > X \). Relief extreme; dispersion resembles horizontal dispersion.

\( \alpha = 2.105(5) \) yellow
\( \beta = 2.32(2) \) yellow \( (2V_e)_{meas} = 92^\circ (-) \)
\( \gamma = 2.65(2) \) orange \( (2V_e)_{calc} = 88^\circ (+) \)

Problems

Send Submissions to:
Lynne A. Price
TUG Macro Coordinator
Calma R&D
212 Gibraltar Dr.
Sunnyvale, CA 94086

Balancing Columns of Text and Translation

In the last issue, Johnny Stovall asked about a macro that could adjust the width of each column of two-column output so that the lengths of the two columns will be equal. His application involves typesetting original texts in parallel with translations. As long as reliable estimates of the relative length of the two segments are available, a simple technique can be used. \protect \texttt{\%varunit} can be set to the width available for text in both columns (excluding margins), and the actual width of each column can then be set in terms of a percentage of \protect \texttt{\%varunit}.

The following macro illustrates this approach using \protect \texttt{\hbox par}:

\begin{verbatim}
\input basic
\%varunit 6in \% Space available for both columns, exclusive of margins
\def\intercolumnwidth{\hskip \hskip .5in)
\% Arguments to \texttt{\trans} are percentages
\% of total width for first column,
\% contents of first column, percentage
\% of total width for second column, and
\% contents of second column, respectively.
\% The two numbers should sum to 100.
\def\trans{\%1\%2\%3\%4\%
\hbox{\hbox par 0.1\%vu
(\%2)\intercolumnwidth\
\hbox par 0.3\%vu(\%4)}
}
\end{verbatim}

Here is a simple example:

Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party. Now is the time for all good men to come to the aid of their party.

And continue ...

Here, it is assumed that there is text extending across the full page width above or below the translations. This macro must be modified if the text segments include multiple paragraphs. A similar macro
could be used to redefine \hspace if the entire text has this form.

Input-Dependent Macro Redefinition

Also in the last issue, Mike Spivak asked about a pair of macros \textit{data} and \textit{list}. Successive calls to \textit{data} would save text that could later be retrieved by calling \textit{list}, so that \textit{list}(1) would produce the parameter passed on the first call to \textit{data}, \textit{list}(2) would produce the parameter passed on the second call to \textit{data}, and so on.

Macros to solve this problem are fairly straightforward—\textit{data} can concatenate its successive arguments on a single string, separating them with some delimiter, and then \textit{list} can pick off the necessary text string. The concatenation of the passed strings, however, is created with \textit{xddef} and a problem arises when some of the arguments themselves contain \textit{xddef}s that the user does not want evaluated until \textit{list} is invoked.

A solution is possible using the technique illustrated by Patrick Milligan's \texttt{DefineFont} macro published in TUGboat, Vol. 2, No. 2. The macro \texttt{macrolist} is defined to contain a sequence of otherwise unused macro names (e.g., \texttt{aa}, \texttt{bb}, \texttt{cc} ...). Each time \texttt{data} is called, it removes the first item from \texttt{macrolist}, saves it on a list \texttt{listitems}, and then defines the specified macro to invoke the argument text. Thus, if \texttt{macrolist} starts \texttt{aa}\texttt{bb}\texttt{cc}..., the first call to \texttt{data} sets \texttt{listitems} to \texttt{aa}, sets \texttt{macrolist} to \texttt{bb}\texttt{cc}\texttt{dd}..., and defines \texttt{aa} to be whatever the argument to \texttt{data} was. The \texttt{xddef} to modify \texttt{listitems} does not cause \texttt{aa} to be expanded, because \texttt{listitems} is redefined before \texttt{aa} is defined. To retrieve the stored macros, \texttt{list} must pick off the appropriate token from \texttt{listitems} and execute it.

The algorithm just sketched is the heart of the macros shown here. The situation is somewhat more complicated, however. \texttt{list} works by using \texttt{xddef}s applied to \texttt{listitems}. To prevent \texttt{list} from expanding any \texttt{xddef}s that might be contained in the argument text \texttt{data} has so far received, \texttt{listitems} is constructed with the undefined macros \texttt{a}, \texttt{b}, \texttt{c}, etc. After \texttt{list} has selected the appropriate item, \texttt{a} is locally defined (i.e., defined within a group) to invoke \texttt{aa}, \texttt{b} to invoke \texttt{bb}, and so on through the ugly macro \texttt{equivalences}. The selected macro is then invoked within the group that defines these equivalences. To allow all this, \texttt{macrolist} is really initialised to \texttt{aa}\texttt{bb}\texttt{cc}\texttt{dd}....

These macros also use the \texttt{Apply}, \texttt{First}, and \texttt{Rest} macros used by Patrick in \texttt{DefineFont}. \texttt{list} uses the recursion macros Brendan McKay described in the same issue of TUGboat. The obvious restriction that these macros are limited to 26 calls to \texttt{data} can, of course, be extended by reinitializing \texttt{macrolist}. A less obvious restriction comes from the local equivalence of \texttt{a} and \texttt{\hspace}. A macro defined with \texttt{xddef} within an argument is usable only from within the same argument, the argument is invoked within a group. \texttt{DefineFont}, of course, have more permanent effects. A restriction can be surmounted by redefining \texttt{Apply} to explicitly test its argument and to call \texttt{aa} argument is 1, \texttt{bb} if it is 2, etc. Unaesthetic as an approach may be, it is not much worse than the current definition of \texttt{equivalences}. Note that "groupless if" such as McKay and Spivak discuss in the last issue must be used to avoid the problem.

\texttt{xddef\listitems}

\begin{verbatim}
\def\data\list{\if !\macrolist{ 
\send{Error: Maximum number of data items exceeded}}}
\else{ 
\Apply\First{\macrolist!} \macro\Rest{\macrolist!} \Remove{code from \macrolist!} 
\Apply\First{\macrolist!} \macro\Rest{\macrolist!} \Remove{code from \macrolist!} 
\macro\Rest{\macrolist!} \Remove{code from \macrolist!} 
\macro\Rest{\macrolist!} \Remove{code from \macrolist!} 
\macro\Rest{\macrolist!} \Remove{code from \macrolist!} 
\Define{macro}\listitems} 
\end{verbatim}

\texttt{\macrolist\#1\#2}

\begin{verbatim}
\let\gdef=\let 
\xddef\Define{gdef \#1} 
\let\gdef=\gdef 
\Define{\#2} 
\end{verbatim}

\texttt{\list\#1\#2}

\begin{verbatim}
\def\temp{\listitems} 
\repeat \#1{times} 
\if !\temp{\setcount{0}{\gdef\macro}}: 
\else{\Apply\First{\temp!} \macro\Rest{\temp!} \{\macro\}} 
\end{verbatim}

\begin{verbatim}
\Apply\First{\temp!} \macro\Rest{\temp!} \{\temp!\endrepeat 
\end{verbatim}

\texttt{\equivalences \macro}

\begin{verbatim}
% The macro \macrolist describes the set of \macro names available to \list and \data
% These macros should never be explicitly \invoked by the user.
\end{verbatim}
A Macro That Prints Its Name

Anthony Siegman asks whether it is possible to devise a \TeX{} macro that will process a one-word argument as both a word of text and a control sequence.

Suppose the argument is entered as \texttt{\{name\}}, or alternatively as \texttt{\{name\} (one or the other, not both). He wants a macro that will:

(a) Accomplish the result \texttt{\{count 4\} such an argument}
(b) Put both the numerical value of \texttt{\{name\} and the}
(c) Send both the numerical value of \texttt{\{name\} and the}

The macro \texttt{\setname} defined below defines a list of its arguments on successive calls, separated by semicolons and a similar list of the values of \texttt{\count4} each time it was called. The macro \texttt{\{name\}} then looks through the list of names until it finds the one matching its argument and returns the corresponding \texttt{\count4} value.

\begin{verbatim}
\def\setname#1#2\{\count4=0 \}
\end{verbatim}

which allows the input
\begin{verbatim}
\setname\{foo\}
\end{verbatim}

to define \texttt{\{name\}} to be \texttt{\{foo\}}. However, despite appearances, \texttt{\findname} has no parameters; \texttt{\{foo\}} is the text following \texttt{\findname}. This technique illustrates that \TeX{} can be made to treat what appears to be a control sequence as two tokens: the text character \texttt{\{\}} followed by a word of text. It can not be extended to solve the complete problem presented by Siegman, since once \TeX{} has determined the character class of a symbol, the character cannot be reinterpreted. Thus, any one occurrence of the four characters \texttt{\{foo\}} can be interpreted either as four text characters or as one control sequence, but can not sequentially be interpreted both ways.

Nevertheless, macros approximating what has been asked for can be written. Instead of defining a macro \texttt{\{foo\}}, the following \texttt{\setname} macro, given the argument \texttt{\{foo\}}, allows another macro to return the value of \texttt{\count4} at the time \texttt{\setname\{foo\}} was called. This new macro is invoked by a single character, \texttt{\{\}, so, in effect, the control sequence defined by \texttt{\setname} is \texttt{\{foo\} rather than \texttt{\{foo\}. Of course, \texttt{\{\@foo\}} is not really a control sequence, but it may be used as such. Actually, \texttt{\{foo\}} is an argument to the macro \texttt{\{\}. \texttt{\{\} expects its parameters to be terminated by a space, the user must remember to include such a space even in situations (e.g., before punctuation) where it would not be required for an actual control sequence.

The \texttt{\setname} macro defined below defines a list of its arguments on successive calls, separated by semicolons and a similar list of the values of \texttt{\count4} each time it was called. The macro \texttt{\{name\}} then looks through the list of names until it finds the one matching its argument and returns the corresponding \texttt{\count4} value.

\begin{verbatim}
\def\setname#1#2\{\count4=0 \}
\end{verbatim}

which allows the input
\begin{verbatim}
\setname\{foo\}
\end{verbatim}

to define \texttt{\{name\}} to be \texttt{\{foo\}}. However, despite appearances, \texttt{\findname} has no parameters; \texttt{\{foo\}} is the text following \texttt{\findname}. This technique illustrates that \TeX{} can be made to treat what appears to be a control sequence as two tokens: the text character \texttt{\{\}} followed by a word of text. It can not be extended to solve the complete problem presented by Siegman, since once \TeX{} has determined the character class of a symbol, the character cannot be reinterpreted. Thus, any one occurrence of the four characters \texttt{\{foo\}} can be interpreted either as four text characters or as one control sequence, but can not sequentially be interpreted both ways.

Nevertheless, macros approximating what has been asked for can be written. Instead of defining a macro \texttt{\{foo\}}, the following \texttt{\setname} macro, given the argument \texttt{\{foo\}}, allows another macro to return the value of \texttt{\count4} at the time \texttt{\setname\{foo\}} was called. This new macro is invoked by a single character, \texttt{\{\}, so, in effect, the control sequence defined by \texttt{\setname} is \texttt{\{foo\} rather than \texttt{\{foo\}. Of course, \texttt{\{\@foo\}} is not really a control sequence, but it may be used as such. Actually, \texttt{\{foo\}} is an argument to the macro \texttt{\{\}. \texttt{\{\} expects its parameters to be terminated by a space, the user must remember to include such a space even in situations (e.g., before punctuation) where it would not be required for an actual control sequence.

The \texttt{\setname} macro defined below defines a list of its arguments on successive calls, separated by semicolons and a similar list of the values of \texttt{\count4} each time it was called. The macro \texttt{\{name\}} then looks through the list of names until it finds the one matching its argument and returns the corresponding \texttt{\count4} value.
TUGboat, Volume 2, No. 3

% Returns first token from list
\def\rest#1,#2!{#2}
% Returns list with first token removed
\def\rest!{#1;#2!#2>

% Allow \name to be invoked by the
% single character @
\def\name{\texttt{\char27\name}}
\chcode`100=13

Testing the Equivalence of Strings
To test the equivalence of strings, the macro \stringeq is called. Shown below, this macro uses the "groupless if" techniques described independently by McKay and Spivak in Vol. 2, No. 2.

% String equivalence--evaluates to T
% if first argument is same string
% as second, F otherwise
\def\stringeq\rest<\rest
\def\stringeqv\rest<\rest

% Main "routine" Save arguments in \stringsa and \stringsb and start testing
\def\stringeq#1,#2!{\def\stringsa<#1
\def\stringsb<#2
\def\endap\endb
\gdef\endbp{\endb}
\gdef\compcharp\compchar
\def\endd\endd}

% Test if at end of first string
\def\endd{\if !\stringsa!\else\endbp
\{\nulltest\}\else\endbp}

% Test if at end of second string
\def\endd{\if !\stringsb!\false
\{\false\}\else\false\endd}

% Compare next characters
\def\compcharp\rest<\rest
\def\compchar\rest<\rest
\def\nulltest\rest<\rest
\def\nulltest\rest<\rest
\def\nulltest\rest<\rest
\def\nulltest\rest<\rest
\def\nulltest\rest<\rest
\def\nulltest\rest<\rest

% When at end of first string,
% test if also at end of second
\def\nulltest{\if !\stringsb!\true\else
\{\false\}

* * * * * * * * * * *

PROBLEMS FROM THE \TeX\XARCHANA CLASS
ANSWERS, AND ANOTHER PROBLEM

These problems are from the video \TeX\Xarchana Class taught by Don Knuth last March.
The solutions given here are based on the one distributed in class. Some parts of the solutions depend on features which, although installed in the current version of \TeX\X at Stanford, may not yet have been implemented elsewhere; an attempt has been made to note such features.

Problem no. 1:
Type:
\vskip 12pt
\noindent\hidei-->Allan Temko

\vskip 2pt
\noindent Architecture Critic

To get:
\-Allan Temko
\Architecture Critic
\def \hidei{\hbox to Opt{\hss \#1}}
Problem no: 2:

Type:
\fancy Senator and Mrs. Stanford had reserved to themselves control of the University's affairs during their lifetimes, including the parceling out of "all the money that could be wisely used." Mrs. Stanford had remained in her husband's shadow—on opening day she could not bring herself to deliver the short speech she had written out. But following the death of the Senator she, at age 65, took on full responsibility for the University with unsuspected strength.

To get:
Senator and Mrs. Stanford had reserved to themselves control of the University's affairs during their lifetimes, including the parceling out of "all the money that could be wisely used." Mrs. Stanford had remained in her husband's shadow—on opening day she could not bring herself to deliver the short speech she had written out. But following the death of the Senator she, at age 65, took on full responsibility for the University with unsuspected strength.

\font H=cmr10 at 30pt
\def \fancy#1\{\hbox{\f\chop to Opt{\hbox{\lower 12pt \hbox{#1}\hskip .1em}}\hangindent \hspace{0 for 2\noindent \hskip \hspace{0 \hskip -.1em}}}\}

The font specified here requires that magnification be implemented; it is also possible to use cmr30, if that is available. (At the AMS, neither is available for the Alphatype, on which this page was prepared, and the large "S" has been obtained from another source and pasted in.)

Note the use of \hide in the last line of this solution; alternatively, this line might begin "\hskip -1wd0 ...".

Problem no: 3:

Type:
\hsize 25em
\noindent This is a case where the name and address fit in nicely with the review. \signed A. Reviewer (Ann Arbor, Mich.)

\vskip 8pt
\noindent But sometimes an extra line must be added. \signed N. Bourbaki (Paris)

To get:
This is a case where the name and address fit in nicely with the review. A. Reviewer (Ann Arbor, Mich.)

But sometimes an extra line must be added.

N. Bourbaki (Paris)
Problem no. 4:
Type:
\point 0 0
\point 1 2
\point 2 1
\point .5 5
\point -1 -1

To get:

* (5,5)

* (1,2)

* (2,1)

* (0,0)

* (-1,-1)

\lineskip Opt
\baselineskip Opt
\varunit .5in
\topspace 3vu \def \point\#1#2 {\vbox to Opt
\vss\vbox to #2vu{
\hbox{\hskip 3vu\hskip #1vu
$\bullet$\bullet \bullet (#1,#2)}\vss}}

Problem no. 5:
Type:
\hspace 20em
\rightjustifythefollowing
This is the first line
{$it This is the second line.}$
{$sl The third.$}
{$bf The last.$}
\endrightjustify
Beginning of another paragraph.

To get:
End of a paragraph.

Beginning of another paragraph.

\def \rightjustifythefollowing{\par
\chcode\='15=12\penalty1000
\vskip-12pt\let\rjm=\rj\rj}
\chcode\='15=12
\def \rj#1
{$rjustline(#1)\rjm}\chcode\='15=5 \%
\def \endrightjustify{\gdef\rjm{\endrj}}
\def \endrj{\chcode\='15=5\penalty1000\vskip-1
Problem no. 6:

Type:
How do you do this?

\$\$\texttt{lineskip 2pt}\$
\texttt{baselineskip 1.3ex}
\texttt{vcenter{\halign{\hfill\hfill\cr
\texttt{linedown}{Look at this \{strange\} pile.}}\quadquad}
\texttt{vcenter{\hfill\hfill\cr
\texttt{linedown}{And at this \{stranger\} one.}}}\$

To get:
How do you do this?

\texttt{linedown}{Look at this \{strange\} pile.}

\texttt{linedown}{And at this \{stranger\} one.}

On the next page appears the “Challenge problem” presented to the \TeXXArcana attendees. The solution will appear in the next issue.
If I have all the eloquence of men or of angels, but speak without love, I am simply a gong booming or a cymbal clashing. If I have the gift of prophecy, and understand all the mysteries there are, and knowing everything, and if I have faith in all its fulness, to move mountains, but without love, then I am nothing at all. If I give away all that I possess, piece by piece, and if I even let them take my body to burn it, but am without love, it will do me no good whatever.

Love is always patient and kind; it is never jealous; love is never boastful or conceited; it is never rude or selfish; it does not take offence, and is not resentful.

Love takes no pleasure in other people's sins but delights in the truth; it is always ready to excuse, to trust, to hope, and to endure whatever comes.

Love does not come to an end. But if there are gifts of prophecy, the time will come when they must fail; or the gift of languages, it will not continue for ever; and knowledge—for this, too, the time will come when it must fail.

For our knowledge is imperfect and our prophesying is imperfect; but once perfection comes, all imperfect things will disappear. When I was a child, I used to talk like a child, and think like a child, and argue like a child, but now I am a man, all childish ways are put behind me. Now we are seeing a dim reflection in a mirror; but then we shall be seeing face to face. The knowledge that I have now is imperfect; but then I shall know as fully as I am known.

In short, there are three things that last: faith, hope and love; and the greatest of these is love.