

T_E^ST Instructions

December, 1990

The purpose this set of macros, hereafter T_E^ST, is to assist in the preparation of multiple choice tests. The first part of the preparation process involves typing in the test. Begin by typing `\problem`, and then type the problem just as you normally would in T_EX. Then you type the multiple-choice answers: proceed each wrong answer with a `\wrong` command and proceed the correct answer with a `\correct` command. If you do not want any multiple choice answers, just don't type any. The problems will be numbered automatically when typeset, but we will also refer to the first problem in the file, the second problem in the file and so on. The file should conclude with an `\end` or `\bye` statement.

There is also a `\note` command: when you have some material which is not a problem but which should appear in the test you type `\note` and then the material. For example, if you have a true-false section, you might want to put a note at the start of this part to warn the students and to tell them how much each of these problems is worth. Notes should not have answers nor will they be numbered in the output, but they do acquire a number based on their order in the file: the first note is note 1, and so on. (Actually, notes can be numbered if you want and indeed the entire labelling process for both note and problems can be controlled by you, the user [see the Extras section below].)

You may have up to five different answers, but you don't need to have that many: just type in as many as you want (up to five), but if you have any answers, exactly one of them must be marked correct. If the problem is a true-false question just type `\Tf` or `\tF` after you are done with the problem. This tells T_E^ST that there are two answers; if you typed `\Tf` then TRUE is the correct answer; if you typed `\tF` then the correct answer is FALSE.

Any time during this process, you may T_EX the document. You will get the problems set in the same order that they occupy in the file, and each problem will have the answers typeset after the problem, again in their natural order, with an underline next to the one that you indicated was the correct one. The notes will also appear at the location that they occupy in the file. The spacing between the lines will be rather cramped, but this will be adjusted later. The purpose of this part is just to get the data entered and to let you print it out for proof-reading. By default, your problems will be T_EX'ed with no magnification at this stage, but, in the final run, magnification will be automatically set to `\magstep1`. (You can change this with the `\setmag` command: `\setmag{1000}` is equivalent to `\magnification 1000` and must be invoked near the beginning of your document. If you do it too late you will get the T_EX incompatible magnifications error. If you want even the proof-reading runs at `\magnification 1200`, just include `\magnification 1200` at the start of your file.)

When you are entering the test, T_E^ST behaves much like T_EX itself, so you should get the desired results easily. One caveat is that braces tend to get removed

in the input process so you should be generous with braces. To make bold, the usual T_EX way to achieve this is to type `\bf` followed by a few braces is dangerous with T_E^ST since the braces will turn the rest of your test turn into bold. The safe way is to use `\bf aardvark }` so that, if a layer of braces is used, both get through, no harm is done.

For people who like to write their own macros, T_E^ST is a bit more difficult and some understanding of how macros work will achieve the desired results. First of all, the `\note` and `\problem` are actually macros and hence may contain other macros. `\note` may not contain: e.g. `\newcount` is `\outer` and `\problem` may contain the first `\note` or `\problem`. All the material between `\begin` commands (or a consecutive `\note` and `\problem` commands, etc.) is read in, processed, and then boxed, and also read in, processed, boxed; and then the boxed material in order and put into the corresponding problem box. Material in the body of the test is nested inside the boxes.

Macros which are defined BEFORE the first run of T_E^ST should work as expected, but any macro defined AFTER should not work as expected. `\gdef` or it probably will not work when you use a constant that you have defined, you should probably use a macro. The constant will not change where you think it should. If you run the file several times (once for each version that it is defined) it will not be read twice and this does not happen to `\note` or `\problem` command. The mechanism of `\gdef` result in a peculiarity: *if you must define something that contains the letter w somewhere in its name, you must not have more than one version of your test at a time.* Also, the `\+` has been set to `\tabalign`. Macros defined AFTER the first run should be designed not to crash when read twice.

T_E^ST does not do too much to plain T_EX and does not load any packages. It does modify plain T_EX's `\shipout` command, so T_E^ST should be loaded after any package which uses `\shipout`. It has not tried to run T_E^ST under AMST_EX, but it should work. A format file in which you have T_EX'ed the T_E^ST macros, L_AT_EX things may be more difficult.

T_E^ST uses plain T_EX's `\headline` and `\footline` commands. `\footline` footlines (particularly to write your own page numbers) should be `\myheadline` and `\myfootline`. See the Extras section.

A final warning: T_E^ST will need to input your test file, which it does using the standard T_EX `\input` command. There are restrictions on file names. The worst restriction is that the file name must NOT contain any SPACES. (The

but spaces are the most common.) For safety, only use the letters a–z, A–Z, or numbers. If you want spaces, use the standard “kludge” of `x_y` for `x y`.

§1. Permutations and typesetting. Once you have the data entered, it is easy to permute either the answers or the problems (or both). Of course you have to describe to `TEST` how you want the permutations done. A *version* of the test consists of the text you typed in plus a choice of ordering for the answers to each problem and a selection and ordering for the problems. For each version, you must describe a selection of problems and notes from your file and in what order you want them to appear in the typeset document. You must also describe a permutation for the answers to each problem for each version.

Each version of your test has a number associated to it, which is used to explain to `TEST` the version to which your permutation data applies. There is a variable, `\firstversion`, which is 1 by default but which you may set to any value you wish. There is also a variable, `\lastversion`, and `TEST` thinks that your versions are numbered consecutively from `\firstversion` up to `\lastversion`. There is probably some limit to the number of versions, but the author has not had the patience to locate it (at this time it is at least greater than 50). (Versions with numbers greater than 100000 are used by `TEST` to indicate it is in a special situation and these numbers should not be used by the casual user.)

By default, `\lastversion` is 0, but when you include the command `\lastversion = n` in your file, you will `TEX` all the versions of the test from `\firstversion` to `\lastversion`. By adjusting `\firstversion` and `\lastversion` you can `TEX` any one version of the test or any range of them.

Version 0 is special (and is the one which occurs if you have not yet included a `\lastversion = n` command). It will give you a printout of the test with problems and answers having the same order as they do in the file, but with the spacing between problems suppressed. Furthermore, the correct answer has an underline next to it. You may also set `\lastversion = -n`: you will get a printout with all the permutations for version *n*, but the spacing will be the usual version 0 “cramped-style”. (This is useful for giving versions to TA’s or colleagues to have them worked.)

In the coming paragraphs, we will describe how to explain to `TEST` how you want the answers for each problem permuted (from their order in the file) for each version and which problems and notes (and in what order) from the file you wish included in each version.

Each multiple-choice problem for a version needs to have a permutation to tell `TEST` in what order to set the answers. Problems which are not multiple choice do not have permutations and neither do problems which are true-false: for these, the choice is always true, then false.

We begin by describing how to permute the answers corresponding to a fixed version, say *v*. The simplest way to do this is to place a permutation command in each problem. If there are 5 answers, a typical permutation command looks like `\permv:baced`, where *v* is the version number. If there are only four answers, you may leave out the *e*, or you may put it last. A command `\perm w:cdeab`, where

w is a number different from *v* will be ignored and will come into its own when typesetting version

When you typeset version *v*, the answers follow from their order in the file as follows: the first will be the old (a); the second will be the old (b); the third will be the old (c); and the last, the old (d). (Recall that you can indicate their natural order, with the correct one indicated by `\lastversion=0` [which is the default].)

The `\perm` command for the problem may be preceded by a `\problem` command which begins the problem and a `\note` command.

There are two additional ways to produce permutations. You may want to declare a global permutation command, `\globalperms{ \permv1:abcde \permv2:cdeba }`, where every problem for version number *v*₁ to *abcde*; and every problem for version number *v*₂ will be *cdeba*. You may use any letters you need and they need not be in any particular order. For answers, you can use all 5 letters and make the permutation for the first 4. This command needs to be placed before the `\note` command. It is most useful if you have an order that you want to typeset. If you want version *v* in natural order, the one command, `\setglobalperms{ \perm1: }`.

Note: All `\perm` commands are just a bit fussy. The *abcde* part may not have any spaces. You may read up to the colon to see if the version number is read in letters, one at a time until it hits a character which point it quits. (This same algorithm applies if you use the `\perm` command because the version for it does not matter.) That will be built is the one which sends ‘a’ to the second; ‘c’ to the third; and so on. If `TEST` quits, the unread ones are sent to themselves. Hence `\permv:baed` is not a permutation since the last answer in the file is not the last answer in the `\perm` command) and as ‘e’ (by default since `\lastversion=0`) is NOT COOL.

You may put a `\perm v:` command in a problem which has precedence over the global one for version number *v*. That way that you can do a global permutation which looks like `\setglobalperms{ \perm1: }` of problems, and then permute the answers for each version.

You may also enter a complete list of permutation commands. The command `\setpermlist{...}` will do this. It should start of the test proper: i.e. before the first `\problem` command. It can be placed in a separate file as long as this file

the start of the test proper. The material between the braces has a rather rigid format. You start with a “`\permv:`” command; on a new line (or at least after a space) type the permutation for problem 1, say *abcde*; on the next line type the permutation for the second problem; etc. until you have one permutation for each problem, one permutation per line. If there are only four answers for a given problem you can either use all five letters with *e* last again or just use a–d. As usual, you may not have any spaces between the letters. In this mode you may include more material after the permutation as long as it contains no spaces and begins with some character other than a–e. This material is ignored by $\text{T}_{\text{E}}^{\text{S}}\text{T}$, but can be helpful to you. For example, you can put the problem number after the permutation: a line which looks like *cdeab17* is a good way to remember that *cdeab* is attached to problem 17. BUT, remember *cdeab 17* is a bad ERROR because of the space. After you have entered the data for one set of permutations, you may enter the data for another or just close the command with a `}`. The `\perm` commands do not need to be in any particular order. You must also remember that true-false questions may not have permutations, but if you create this file by just working your way through a copy of the test (with the problems in their natural order) you will generate the correct file. There are also other programs which you can use to generate a file of random permutations which you can easily edit to be fodder for this macro.

If you have two (or more) versions with the same permutations for the answers, you may type “`\perm v1&v2:`” and then the list: this is equivalent to “`\perm v1:`” and the list followed by “`\perm v2:`” and the same list.

You may still put a “`\perm v:`” command in a problem and it will take precedence over the one for version number *v* set from the list. It seems rare that you would want to do this and you still must have an entry in the list for the problem even though it is to be ignored. (It is occasionally useful for trying out a new permutation before putting it in the list.) You may also use a different method to enter the permutations for different versions: the method for one version could be global; the method for another could be from a list; and the method for a third could be from `\perm` commands after each problem.

If $\text{T}_{\text{E}}^{\text{S}}\text{T}$ encounters a multiple-choice problem for which it can find no permutation for the answers, it displays a message, uses *abcde* as a default permutation, and continues. You will need to re- $\text{T}_{\text{E}}^{\text{X}}$ the file after entering the desired permutation, but this error will not mess up the run too badly so usually you may as well finish and see if you have forgotten more than one permutation.

There are two other errors associated with multiple-choice answers. If no `\correct` answer is given for a problem, $\text{T}_{\text{E}}^{\text{S}}\text{T}$ complains by writing a note to the log file for you. This error is annoying, but will not mess up your run too badly: $\text{T}_{\text{E}}^{\text{S}}\text{T}$ assumes that the correct answer is the first one in your file and goes on (this can be serious if you believe the subsequent marked answer sheet). A more serious problem is to have two (or more) `\correct` answers. Again $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will complain in the log file. If you really have just marked two answers correct, then this is not so serious, but the most common cause of this error is to forget a

`\problem` command. $\text{T}_{\text{E}}^{\text{S}}\text{T}$ now thinks your problem is a problem which will look strange, but more seriously it means that the bets for this problem. This is too many; $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will do what the bets as to subsequent behavior are off.

Once the permutation data for the answers is entered, you may modify the order of the problems. If you wish to reorder the problems in the same order as in the file, then you may use the

You may also prepare versions of the test using the `\setproblempermutations{...}` command. This command has the following format. Begin with a “`\version v`” command reordering for version *v*. Next comes a stream of permutation data that it has saved. The entries in this list are of the form `number` means to unbox the problem which has the permutation `number` are two other possible entries in this stream: a `pagebreak` (which is important) will force a page break at this point. `number x` at this point. You need not use the `pagebreak` command runs through your list doing the unboxing and pagebreaking. It quits when it reaches the end of the list. (Perhaps `pagebreak` is not the best name for this command since it also unboxes problems as well as permute them.)

If you want the same problem list for two (or more) versions, use “`\version v1&v2:`” and then the list.

When you are typesetting a version of the test, you can force a page break with the `\pagebreak` command. Do not break in the middle of a note or problem. Here `\pagebreak` after the note or problem in which the `\pagebreak` command cause the next problem or note to appear at the top of the typesetting version *v*, and there is a `\version v` command. `s`, any `\pagebreak` commands in the file are ignored. `s` generated from the version information as discussed above.

Of course $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will also insert page breaks in the material to fit on a page (except that break in the middle of notes or problems). (If you have hired Thomas A. Knuth, you can imitate a page break in the middle of a problem part of the problem as a note; include a `\pagelabel` command to finish the problem as a problem. [After reading the `\pagelabel` label your note with `\global\notelabel={{\count0 by1 \hss\number \count0.\ \ }}\ \ }` in the log file. labels problems. Be sure to kill `\notelabel` before the next `\global\notelabel={}.]`)

$\text{T}_{\text{E}}^{\text{S}}\text{T}$ takes the extra space on a page and sets each page. It does this by setting the page number containing each problem. None of the extra space

we will explain in the section on spacing, you can specify the minimum amount of space after a problem. One way to handle the page breaks is to just put the minimum space you want after each problem and let $\text{T}_{\text{E}}^{\text{S}}\text{T}$ put the breaks where it wants. It is not necessary to have “pb.”’s if you like $\text{T}_{\text{E}}^{\text{S}}\text{T}$ ’s break points.

A common error in both answer and problem permutations is to include the same answer or problem twice. $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will warn you that this has happened with an “Empty $\left\{ \begin{array}{l} \text{answer} \\ \text{problem box!} \\ \text{note} \end{array} \right.$ ” message. In the typeset document the repetition will result in a blank box corresponding to the location of the second (and any subsequent) usage.

§2. Answer Sheets. $\text{T}_{\text{E}}^{\text{S}}\text{T}$ also generates an answer sheet for the test. You should have a $\backslash\text{title}$ {Math 999} and a $\backslash\text{date}$ {December 25} command near the start of your file. This will put the title and date on the copy used for proof-reading and also puts this title and date on the answer sheet. (A $\backslash\text{twolinetitle}$ command is available if you need two lines: it needs two variables, the first line and then the second line. There is also a $\backslash\text{comment}$ {whatever} command which adds a comment line just under the date and an $\backslash\text{answersheetfootline}$ {whatever} command which adds the material to the bottom of the page. Most of the answer sheet is taken up with lines for the students to mark their answers, but we also include a line for their name, a line for you to record their score, and optional lines for your name and their section.

The section number line can be suppressed by the $\backslash\text{nosection}$ command and the line for your name can be suppressed by the $\backslash\text{noprofessor}$ command. (The $\backslash\text{Professor}$ {Professor Hilbert} command will add the professor line but fills it in with “Professor Hilbert”).

For each version of the test, $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will generate an answer sheet with the correct answer marked with a black dot. This sheet comes at the end of the test questions for that version. It is labeled with a version number and lists how many answers were a , how many were b , etc.

Immediately after the marked answer sheet for version 1 is produced, $\text{T}_{\text{E}}^{\text{S}}\text{T}$ also produces an unmarked answer sheet. This unmarked sheet also includes your footline text, which is suppressed in the marked versions because that space is used for the answer information. Sometimes, because of the way you have permuted the problems, the answer sheets for different versions ought to look different. Whenever this happens, $\text{T}_{\text{E}}^{\text{S}}\text{T}$ produces a new unmarked answer sheet immediately after the corresponding marked one. Whenever an unmarked answer sheet is produced $\text{T}_{\text{E}}^{\text{S}}\text{T}$ also writes a “UNMARKED ANSWER SHEET” to the log file.

The answer sheet feature can be suppressed with the $\backslash\text{noanswersheet}$ command. If this command is included in your file, then $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will not produce any answer sheets, but it will generate a file which contains the information as to how the tests were typeset and what the correct answers are. This file is called $\backslash\text{jobname.answer}$:i.e. if the file containing your test is called My_Test or My_Test.tex ,

this file will be called My_Test.answer . (You names, didn’t you?) You can process this file if swer sheets. The format of this file is described also use the file to produce the standard answer $\backslash\text{writeanswersheet}\#1$ command.

To do this, create a separate file with your answer sheets as usual and finish the file with $\backslash\text{writeanswersheet}\#1$ (NOT $\backslash\text{bye}$). The file “file name” should be the name of the file you are typesetting the tests with the $\backslash\text{noanswersheet}$ command. Do not use spaces or other weird characters in the file name. The file name you like [modulo your machine’s limitations] will still be “something.answer”.) If you prefer to use $\text{T}_{\text{E}}^{\text{S}}\text{T}$ directly. Add your formatting commands to the end of this file (the material that is in the file when you run $\backslash\text{writeanswersheets}$ just before the original material). The command $\backslash\text{writeanswersheets}$ will not work if you use $\backslash\text{end}$ before the $\backslash\text{end}$ (which cannot be replaced with anything).

By default, the program puts a line with a “1” at the end of page $n-1$ and the first problem on page n (for marking the wrong numbered problem. This can be suppressed or suppressed altogether. You can also get it to be “1” if you wish. These changes are accomplished with the $\backslash\text{pageannouncements}$ command. $\backslash\text{pageannouncements}$ { xxx }, where xxx can be “none” (type {none} (again no spaces), or “withfirstpage”, or “usual” (type {usual} get no extra lines at all; if it is “blank” then you get no extra lines at all; if it is “blank” then you get the effect described above (and the $\backslash\text{pageannouncements}$ command at all). The “1” line. If you use anything else, you will get a different default.

By default, these page announcements are in the right margin, but you can change this if you wish. The $\backslash\text{mymargin}$ macro, $\backslash\text{mymargin}$, which you may use in your file. The default is $\backslash\text{def}\backslash\text{mymargin}\{\text{Page}\}$. The $\backslash\text{pagecounter}$ is a variable you may use in your file. To move the announcements to the left hand edge of the page use the following definition in your file: $\backslash\text{def}\backslash\text{mymargin}\{\text{Page}\}$. The $\backslash\text{answersheetlinewidth}$ macro, $\backslash\text{answersheetlinewidth}$ { Page $\backslash\text{number}$ $\backslash\text{pagecounter}$ }, which you may use in your file. The $\backslash\text{answersheetlinewidth}$ is a variable which you may use in your file.

It is possible to put the data into two columns. $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will do this: the format is $\backslash\text{twocolumn}$ { b } where b is the number of columns to do the break: if you put in a number for b , then $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will do the break at the top of the second column. If you put in “blank” the line with “Page r ” will appear at the top of the page. If you are suppressing page announcements then the next

in “n.r”, then the first problem or page break after note r will appear at the top of the second column.

It is also possible to select the *style* for the entries in the answer sheet as well as the labels in the main body of the test. We discuss this in the next section.

The answer sheet is composed of three boxes and material at the bottom of the page. The location of these items on the page can be adjusted. The first box is the header box which contains the title, date, any comment, the area for the student’s name, professor, and section number. (This box is actually named `\header` and you may use it in your own macros if you wish.) It is set so that its upper right corner has the coordinates specified by `\righthead` and `\vertthead`. By default these are 0in. and -.5in. respectively, but can be adjusted by the user at will with a `\global\righthead={whatever}` command. A second box contains the lines for you to record the student’s score. It can be adjusted by changing the `\righttotals` and `\verttotals` dimensions, and the name of this box is `\totals`.

There is another box, the multiple-choice box, which contains the area for the students to mark their answers. The location of this box can be adjusted by changing the `\rightmultchoice` and `\vertmultchoice` dimensions. The defaults here are 0in., .8in. but the best way to adjust any of these boxes is with the `\global\advance` command. For instance “`\global\advance\vertmultchoiceby 1in`” will lower the multiple-choice box by 1in. on the page. The name of this box is `\abcdebox`.

Warning: Since the file is read several times, `\advance`’s should be used cautiously. An `\advance` like that suggested above should only appear before the first `\note` or `\problem` command or on subsequent answer sheets the multiple-choice box will migrate steadily down the page.

By default, the material at the bottom of the answer sheet is centered. If you prefer some other convention, `\skipforfootlineofanswersheet` can adjust this for you. For example, `\skipforfootlineofanswersheet=0pt` will left justify the material.

The length of a line of multiple choice answers is set by `\answersheetlinewidth` and can be adjusted by setting it with a `\global` command or changing it with a `\global\advance` command. The space after the problem number and before the first answer is set with `\problemnameskip`. In two column mode the length of a line of answers is roughly half of `\answersheetlinewidth` and the two columns are separated by a space of `\columnspace`. The space between each line of answers is set with `\multiplechoiceskip`. The space between the labels ((a), (b),etc. by default) is simply set using `\hfil`’s between the labels. You can also write your own problem numbering schemes as described in the Extras section below.

Finally, if all this is not enough to produce the perfect answer sheet, you may define your own routines. You may define a new command, `\mymarkedanswersheet`, to produce marked answer sheets and `\myunmarkedanswersheet` to produce unmarked answer sheets. \TeX will call your commands rather than

processing the data as above. Your routines should use `\placeheader` (for the header pages) and `\eject` them.

There are three commands, `\placeheader`, `\multchoicebox` which you can use in your macros, `\def\mymarkedanswersheet#1{\placeheader#1\multchoicebox\vfil\eject}\let\myunmarkedanswersheet=\mymarkedanswersheet` is how the usual answer sheet is done. It gives you easy access to the output routines for the answer sheet through the “everyanswersheet” token list since if you really want to use the `\mymarkedanswersheet` code. As a further example, `\def\mymarkedanswersheet#1{\setbox\header\material}\box\header\placetotalsbox\placemultchoicebox\myunmarkedanswersheet=\mymarkedanswersheet` will produce an answer sheet with your header material in the header box. The difference between marked and unmarked answer sheets is that \TeX passes the `\abcdebox` with the answers marked and the `\abcdebox` is unmarked.

§3. Label Styles.Both in the answer sheet and in the test itself, multiple choice answers. By default these are the letters A-E. This can be changed to any other scheme the user likes. For example, to do this: `\setlabels ABCDE` will change them to A-E. This will produce a scheme no one can remember except by looking at the answer permutations as through the labels in your chosen labels.

Each label has a style (e.g. italics, or roman) and the style for the labels in the test itself is set with `\let\labelstyle=\sl` or whatever. (The default is a serif font.) The styles are macros, and if you are familiar with them you can even write your own label styles.

Each label also has a border: to get an (a) style label is a pair of parentheses. This can be set with `\border=\parenthborder`, which is the default. `\border=\boxborder`, which is also the default, places a square border. `\border=\boxborder` which places a square border. `\boxwidth` which is dropped a distance of `\boxwidth` and `\boxdepth` can be adjusted with the `\advance` command.) The user can also create their own border commands. `\border#1{{#1}}` will create [a] as a border. A macro `\magcmsy=cmsy10 scaled 1200\def\labelborder#1{\magcmsy\hss\bigcirc\textfont2\hss#1\hss}` (where the `\font` command is a “font”) will produce a-e with circles around them:e.g. (

§4. **Problem Spacing.** \TeX has a large number of spacing conventions which \TeXST has partially disabled to get things to work well. \TeXST has certain spacing parameters of its own in order to allow you to get the spacing you want. Descriptions of spacing parameters for the answer sheets occur in the Answer Sheet section above.

The body of a problem has a certain interline spacing. The \TeX command $\text{\lineskip} = 4\text{pt}$ will set this to 4pt. for the problem being set when it was invoked. This command in a note will set the interline spacing for the note. Without a \global command, the result is local to the problem or note. You can use this to open up (or tighten) the spacing between the lines of a problem or note. If you want the interline spacing for all problems to be 4pt. (but you do not want to make the change global because you do not like its effect on your notes or your answers, you can say $\text{\everyproblem}\{\text{\lineskip} = 4\text{pt}\}$. (See the description of \everyproblem and \everynote in the Extras section.)

In the problem a small amount of math-surround often improves the display of the problem, but it rarely improves the display of the answers, so we do not want to change \mathsurround globally. (Math-surround is space that \TeX inserts around math mode stuff.) The plain \TeX command $\text{\mathsurround} = 2\text{pt}$ will set the math-surround to 2pt.'s in the problem with the change being local to the problem. (\everyproblem can be used to make a change in every problem.)

Perhaps here is also the place to remark that \everymath has been set to \displaystyle .

The most common spacing we wish to adjust is to set the minimum space after a problem. The commands \afterproblemskip and $\text{\normalafterproblemskip}$ do this: " $\text{\afterproblemskip} = 1\text{in}$ " puts a minimum of one inch of space after the problem in which it is invoked; $\text{\normalafterproblemskip} = 1\text{in}$ puts a minimum of one inch of space after every problem beginning with the next one. (There are also a $\text{\normalafternoteskip}$ and an \afternoteskip .)

When there is more than one line of answers for a problem, you can control the space between lines with the \answerlineskip and $\text{\normalanswerlineskip}$ commands. As usual, the first is local to the problem and the second begins with the next problem.

The commands $\text{\answerskip}\#1\text{before}\#2\text{after}$ and $\text{\normalanswerskip}\#1\text{before}\#2\text{after}$ put some space between the label and the start of the answer (the dimension #1) and it insures that there is a certain minimum space after the answer (the dimension #2).

Note: These skips are true \TeX skips, but because they are deeply buried, they behave much like dimensions. An $\text{\afternoteskip} = 1\text{in}$. will put an inch of space after the note, but $\text{\afternoteskip} = 0\text{pt}$ plus 1fil will do nothing because the glue is set long before the box containing the note is put into the main vertical list.

§5. **Extras.** You have four choices of how the These choices are selected by setting the variab 0 you get no page numbers at all (this can also **numbers**). The number 1 is the default and yo hand corner, except for the first page, which has same except that the first page also has a num all the pages at the bottom with the numbered do the same except that the first page has no n set with a $\text{\global}\text{\pagenumberstyle} = r$, and effect at the next \shipout , which, given \TeX command or, perhaps the previous page.

There is a token, \everyversion , which v ginning to process the text for each version. $\text{\ifodd}\text{\version}\text{\relax}\text{\global}\text{\pagenumberstyle} = 4\text{\fi}$ will alternate the location of th This example also demonstrates that \version i the user, which holds the version number of the

\everyversion is expanded just before the c is assembled. Hence it should be possible to a \everyversion and have things work out correc which skips certain versions (1 and 3) and does below). The specifics of \killversion come fro roughly, we skip over the code which produces ease back into the loop which is working throug \killversion does the skip; the \fi is to close off \fi is part of the code we skipped); and the \global we skipped from \killversion but that we real $\text{\version}\#1\text{\global}\#2\text{\global}\{\text{\fi}\text{\global}\}\text{\every}\text{\killversion}\text{\fi}$

There are also \everyproblem and \every just before processing each problem or each no mands after the test has begun, but then the preceded by \global before they will have any \everyproblem are expanded at the start of a defined sometime before you want them. (If \global text of a problem, it will not take effect until th

There is also a command $\text{\ifversion}\#1:\#2\&\#3\&\#5:$ $\text{\afterproblemskip} = 1\text{in}\text{\fi}$ is place in versions 2, 3 and 5, the \afterproblemskip will be whatever $\text{\normalafterproblemskip}$ is command, but if you put $\text{\ifversion}2\&3\&5:\text{\fi}$ in your file, this will set \afterproblemskip and 5. This command is intended to be used \ifversion command is expanded as it is read,

`\problem` or `\note` command it is only read once and hence will have little effect. If you need it executed before the first `\note` or `\problem`, just put it in `\everyversion`.

Finally there is a token list `\everyanswerbox`. This token is expanded just before the box containing the answers is attached to the box containing the problem. The default for this token list is `\everyanswerbox ={\ifnum\lastversion < 1\relax\vskip 4pt \else \vskip 12pt\fi}`. (This routine puts 4pt.'s of space between the bottom of the problem and the top of the answers in “cramped–space” runs and 12pt.'s of space otherwise.) The command immediately following `\everyanswerbox` is `\box` followed by the number of the box containing the answers. You could write a command, `\def\afterbox\box#1{ something }` which could process this box: `\def\afterbox\box#1{\vskip 12pt \box#1 }` `\everyanswerbox ={\afterbox}` is equivalent to the usual routine except that it always puts 12pt.'s of space between the problem and the answers; `\def\afterbox\box#1{\vskip 12pt \hbox to\hsize{\hskip 1cm \box#1\hss}}` `\everyanswerbox ={\afterbox}` will shift all the answer boxes 1 cm. to the right. \TeX has also stored the individual boxes with the answers which are still available to you (see the section on “Answer Typesetting” below).

The numbering of the problems is set by a token list, `\problemlabel`. By default, this is `\problemlabel={\number\probcounter.\ \ }`. These tokens precede the problem and extend into the left margin (the end of the list is the beginning of the left margin). There is also a token list, `\answerlabel`, which is used to label the answer sheet. The same expansion scheme applies: the right end of the list is fixed and it extends as needed to the left. The commands `\problemlabel={\ifnum\probcounter < 11\relax A:\number\probcounter.\else B:\count0=\probcounter \advance\count0 by -10\number\count0.\fi}` will precede problems 1–10 with “A:”; problems 11 on will be numbered as “B:1.”, B:2.”,etc. If you add `\let\answerlabel =\problemlabel` then this style will also be adopted for the answer sheet, but this is up to you.

There is also a token list, `\notelabel`, which puts a label before each note, extending into the left margin just like `\problemlabel`. By default this token list is empty, but it can be set to anything you want.

Both the plain \TeX `\headline` and `\footline` token lists are managed by \TeX and so are not available to the user. As indicated in the introduction, we have supplied `\myheadline` and `\myfootline` to replace them. If you define a headline or a footline token list using `\myheadline` or `\myfootline`, your material replaces \TeX 's material and is centered automatically using `\hfil`'s (so you can left or right justify using `\hfill`'s). The command, `\myfootline={Footlines\hfill}`, will put “Footlines” at the base of every page, beginning at the left margin. It will also kill page numbers in styles 2 and 4 since the material in `\footline` is now yours and not \TeX 's. At least one use for these commands is to produce other page numbering schemes. At `\shipout` time, the variable `\pageno` contains the page number of the page being shipped out. Hence `\myfootline={\ifodd\version \number\pageno.\else \number\pageno\fi}` `\nopagenumbers` will produce

a scheme in which page numbers are centered at periods after them in odd numbered versions and even versions.

The command `\sevenrm` is available to pr

There is a command, `\pftreadremark#1`, you put in a remark to yourself or your coauth test. Ideally you should remove these before ma forget, these remarks do not print in versions wit send a “YOU STILL HAVE A REMARK IN TH The “Problem . . .” messages that you have notic the problem is finished, so if your remark is in a p the “YOU STILL HAVE A REMARK IN THE I `\pftreadnote` is a macro, so your text needs to

There are two commands for producing frac sets a fraction with #1 in the numerator and #2 with the numerator and denominator in plain \TeX than the usual default in plain \TeX for setting f `frac#1#2` which also sets a fraction but this ti conventions.

Two commands which are of use to Textures (`#3`) and `\scaledpicture#1by#2(#3scaled)` of these can be found in the Textures manual, picture in the Textures “pictures” window, #1 listed in the window of the picture you want. Th \TeX scale factor (e.g. 500, 1000,1200,etc.) The r is a picture contained in a box which just encl `\picture#1by#2(#3)` will raise the picture

§6. Answer Typesetting. Normally you sho typesets your answers, but occasionally you m effect. This section explains how \TeX preforms with it.

Each answer is boxed up as it is read. At the the correct permutation to use, adds labels to th boxes into box registers `\bba`, `\bbb`, `\bbc`, `\bbd` label (a) is in box `\bba`, etc. The answers, but box registers `\ba`, `\bb`,`\bc`, `\bd` and `\be`. There which hold the position of the correct answer (0

Then \TeX determines how to place these l which \TeX may use. An imposed requirement is one line of answers to the next is never increas

We may be able to place all our answers o does this is called `\fivezero`. Or we could pl the next if needed: this command is called `\fo predilection for setting the answers as three on`

this is also possible. There is a $\text{T}_{\text{E}}\text{X}$ `\if` command, `\iftruefourone` which does this. The default is `\truefouronefalse` but if you say `\global\truefouronetrue`, thereafter, $\text{T}_{\text{E}}\text{S}\text{T}$ will set all your “four-one”’s as four answers on the first line and one on the second.)

There is a command, `\threetwo`, which sets three answers on one line and tries to set two on the next: if called when this is not possible, it defaults to `\threeone`, which sets three answers on one line and one on each of the next two lines.

There is a $\text{T}_{\text{E}}\text{X}$ `\if` command, `\ifthreetwoB`, available to the user. There are two ways we could get three answers on line one and two answers on line two. By preference, $\text{T}_{\text{E}}\text{S}\text{T}$ will try to begin the second answer, (f), on line two under the second answer, (b), on line one (case A): if it can not do this because answer (d) is too long, $\text{T}_{\text{E}}\text{S}\text{T}$ will try to begin answer (f) under the third answer, (c), on line one (case B). Some people do not like this second arrangement and would prefer that the answers appeared on three lines with one answer each on the last two lines rather than case B of “three-two”. The command `\threetwoBfalse` will do this. By default, we have `\threetwoBtrue`, and with a `\global` command this can be turned on and off during the run to force some answers as in “three-two-case B” and others to set as “three-one-one”.

There are commands, `\twotwo`, which tries to set two lines of two answers each and a third line of one, which defaults to `\twoone` when it can not do this: `\twoone` sets two answers on line one and one answer per line afterwards.

Finally, `\onezero` just sets each answer on a line by itself.

These routines are semi-available to the user. First note that it is the users responsibility to see to it that the requisite number of answers will fit on the first line (else you will get an “overfull hbox” message). After the first line, the routines manage the remaining ones. By saying `\global\everyanswerbox = {\vskip 14pt\answerbox} \gdef\answerbox\box#1{\threeone\box#1}`, all problems after this command will have their answers set in “three-one-one” format. This will surely not work well in general, but by changing `\everyanswerbox` before a problem ends and then restoring it to its old value in the next problem, one can force the answers of a particular problem to be set in “three-one-one” style when $\text{T}_{\text{E}}\text{S}\text{T}$ might naturally set it in “three-two” style. To facilitate such changes, we have defined a `\normaleveryanswerbox` which is the same as default `\everyanswerbox`: hence `\global\everyanswerbox = \normaleveryanswerbox` will restore `\everyanswerbox` to its default value (unless the user has also redefined `\normaleveryanswerbox`).

There is a routine, `\handbreak#1`, which will set a problem: if you wish every version of a particular problem set as, say `\threetwo`, just put `\handbreak\threetwo` into that problem and it will happen. If you only want this for certain versions, put in, for example, `\handbreak{\ifversion 1&3:\relax\threetwo\fi}`. In general, the variable, `#1` from above, should be one of the line setting routines discussed above.

§A.1 Answer File format.

If there is a `\noanswersheet` command, $\text{T}_{\text{E}}\text{S}\text{T}$ will use `\jobname.answer`.

The first entry in this file is **Version**, followed usually by 1 and a `\par`. Then comes a list of entries, either the file ends or we encounter another **Version** entry, followed by the next version number.

These additional entries are of one of five types:

1. As a note is put into the main vertical list, $\text{T}_{\text{E}}\text{S}\text{T}$ adds a “n.x” line to the file, where *x* is the number of this note in the version of the test being printed.
2. As a problem with no multiple-choice answers is put into the main vertical list, it adds a “P.x” line to the file, where *x* is the problem number, *y* is a number in the range 0 to 4 indicating the version of the test being printed, and *z* is the number of multiple choice answers.
3. As a multiple-choice problem is put into the main vertical list, it adds a “t.x.y.z” line to the file, where *x* is the problem number, *y* is a number in the range 0 to 4 indicating the version of the test being printed, and *z* is the number of multiple choice answers.
4. A true-false problem puts a “t.x” line into the main vertical list, where *x* is the problem number, 0 if TRUE is the correct answer and 1 if FALSE is the correct answer.
5. The fifth type of entry is generated using `\markdata`. If an item of type 1–4 is added to the file, $\text{T}_{\text{E}}\text{S}\text{T}$ adds a “n.x” line to the file, where *n* is the same data. Then, as a page is shipped out, $\text{T}_{\text{E}}\text{S}\text{T}$ adds the mark data to the file. (If the last item on the page is of type 1–4, there is an “n.x” in the file and the file will be broken up into two pages. It is usual with $\text{T}_{\text{E}}\text{X}$, the page break is often not at the end of the page or two too many into the main vertical list.)

The mark data is not emitted until after the `\endmarkdata` command. Your `\everyanswerbox` routine can manage the mark data for types 1, 2, 3, and 4 as well.

If $\text{T}_{\text{E}}\text{S}\text{T}$ is making the answer sheets, this data is saved. It is important that, if $\text{T}_{\text{E}}\text{S}\text{T}$ is to generate the answer sheets, the mark data be correct and be what $\text{T}_{\text{E}}\text{S}\text{T}$ expects. (In the `\jobname.answer` file, you can arrange the mark data for items of type 3 and 4.) Most uses of `\everyanswerbox` will set the problem number or the number of answers or the number of multiple choice answers usually the mark data is correct without any further action. If the problem number is `\probcounter`, the correct answer is `\correct`, and the number of answers is `\numberps`.

§A.2 Some technical data. The routines and variables listed below represent the commands and variables used in the answer sheets. THIS IS NOT YET CORRECT (with some effect, you shouldn’t)! There are however a great many other routines and variables in $\text{T}_{\text{E}}\text{S}\text{T}$. They have been protected from the casual user.

of their name after first declaring ‘?’ to have category code 11 at the start of $\text{T}_{\text{E}}^{\text{S}}\text{T}$. Of course we redeclare the category code of ‘?’ to be 12 at the end of the macros for $\text{T}_{\text{E}}^{\text{S}}\text{T}$, so, under normal circumstances, the user may write and use other macros freely with $\text{T}_{\text{E}}^{\text{S}}\text{T}$.

$\text{T}_{\text{E}}^{\text{S}}\text{T}$ also plays games with the category code of the letter w. $\text{T}_{\text{E}}^{\text{S}}\text{T}$ needs to read the file once for each version of the test desired and the material before the first note or problem should not be read twice just in case the user has defined some macro which will not do well when read a second time. It is easy enough to skip over material at the start of the test the second time the file is $\backslash\text{input}$ UNLESS some of it has been declared $\backslash\text{outer}$. The most common situation in which this occurs in $\text{T}_{\text{E}}^{\text{X}}$ is with the various $\backslash\text{new}$ commands. Hence, by changing the category code of w whenever we start to re-read the file, $\text{T}_{\text{E}}^{\text{S}}\text{T}$ glides harmlessly past some $\backslash\text{ne}$ commands until it hits a $\backslash\text{note}$ or $\backslash\text{problem}$ command, at which point the category code of w is restored. We have also redeclared the plain $\text{T}_{\text{E}}^{\text{X}}$ alignment command, $\backslash+$ (which is $\backslash\text{outer}$), to be $\backslash\text{tabalign}$. Furthermore, $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will also pass harmlessly over your $\backslash\text{outer}$ macros if they contain a w, but otherwise it will halt with an error message, your macro will be turned into mush, and the whole project should probably be abandoned. (If you have some macros which should be read every time a new version starts up, you can include them in the $\backslash\text{everyversion}$ token list.)

The advantages of reading the file each time far outweigh these minor annoyances. It means that we need only a minimum amount of storage so $\text{T}_{\text{E}}^{\text{S}}\text{T}$ rarely has memory problems. The worst case from the point of view of memory usage occurs when we $\text{T}_{\text{E}}^{\text{X}}$ a version in which we permute the problems. For this case we have adopted a compromise strategy: $\text{T}_{\text{E}}^{\text{S}}\text{T}$ reads the file only once, but only stores those problems and notes that it is going to need to assemble the test. Furthermore, $\text{T}_{\text{E}}^{\text{S}}\text{T}$ only reads the file until it has found all the needed material, after which, it stops reading. $\text{T}_{\text{E}}^{\text{S}}\text{T}$ could have $\text{T}_{\text{E}}^{\text{X}}$ 'ed much larger tests by reading the file to find a needed item and then reading it again to find the next one, but this seemed to be sufficiently slow that it was not implemented. (It also seems rare that one wants a hundred question test with the problems permuted, and if you do then perhaps $\text{T}_{\text{E}}^{\text{S}}\text{T}$ is not the tool to generate the initial file. Once you have the file with the problems and notes in the order you wish, $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will $\text{T}_{\text{E}}^{\text{X}}$ the file, essentially regardless of its size. Of course you had better shut off the answer sheet macros and produce the sheets yourself or you will overflow $\text{T}_{\text{E}}^{\text{X}}$'s memory.) A practical limit to the size test $\text{T}_{\text{E}}^{\text{S}}\text{T}$ will accommodate is about 50 problems. (After this, the answer sheet macros cannot fit the stuff on one page so you will certainly have to make your own answer sheets. If you select more than 50 problems, memory begins to get a little tight: the author has $\text{T}_{\text{E}}^{\text{X}}$ 'ed 50 a problem test with $\backslash\text{tracingstats}=2$ and observed about 6000 words of memory were still untouched. With 55 problems, we had 499 word of memory still untouched. Your statistics will vary depending on how complicated your problems and answers are.)

$\backslash\text{abcdebox} \dots \S 2$
 $\backslash\text{afternoteskip} \dots \S 4$
 $\backslash\text{afterproblemskip} \dots \S 4$
 $\backslash\text{ansborder} \dots \S 3$
 $\backslash\text{ansstyle} \dots \S 3$
 $\backslash\text{answerlabel} \dots \S 5$
 $\backslash\text{answerlineskip} \dots \S 4$
 $\backslash\text{answersheetfootline}\#1 \dots \S 2$
 $\backslash\text{answersheetlinewidth} \dots \S 2$
 $\backslash\text{answerskips}\#1\text{before}\#2\text{after} \dots \S 4$

 $\backslash\text{ba}\text{--}\backslash\text{be} \dots \S 6$
 $\backslash\text{bba}\text{--}\backslash\text{bbe} \dots \S 6$
 $\backslash\text{boxborder}\#1 \dots \S 3$
 $\backslash\text{boxdepth} \dots \S 3$
 $\backslash\text{boxwidth} \dots \S 3$

 $\backslash\text{columnspace} \dots \S 2$
 $\backslash\text{comment}\#1 \dots \S 2$
 $\backslash\text{correct} \dots \textit{Introduction}$
 $\backslash\text{correctcounter} \dots \S 6$

 $\backslash\text{date}\#1 \dots \S 2$

 $\backslash\text{end} \dots \textit{Introduction}$
 $\backslash\text{everymath} \dots \S 4$
 $\backslash\text{everynote} \dots \S 5$
 $\backslash\text{everyproblem} \dots \S 5$
 $\backslash\text{everyversion} \dots \S 5$

 $\backslash\text{firstversion} \dots \S 1$
 $\backslash\text{fivezero} \dots \S 6$
 $\backslash\text{fourone} \dots \S 6$
 $\backslash\text{frac}\#1\#2 \dots \S 5$

 $\backslash\text{header} \dots \S 2$

 $\backslash\text{ifthree} \text{twoB} \dots \S 6$
 $\backslash\text{ifturefourone} \dots \S 6$
 $\backslash\text{ifversion}\#1:\#2 \dots \S 5$

 $\backslash\text{labelborder} \dots \S 3$
 $\backslash\text{labelstyle} \dots \S 3$
 $\backslash\text{lastversion} \dots \S 1$

 $\backslash\text{multiplechoiceskip} \dots \S 2$
 $\backslash\text{myfootline} \dots \S 5$
 $\backslash\text{myheadline} \dots \S 5$
 $\backslash\text{mymarkedanswersheet} \dots \S 2$

Pseudo-Index

`\mypageannouncement#1` §3
`\myunmarkedanswersheet` §2

`\noanswersheet` §2
`\noprofessor` §2
`\normalafternoteskip` §4
`\normalafterproblemskip` §4
`\normalanswerlineskip` §4
`\normalanswerskips#1before#2after` §4
`\nosection` §2
`\note` *Introduction*
`\noteline` §5
`\notelineskip` §4
`\numberps` §6

`\onezero` §6

`\pageannouncements=#1` §2
`\pagebreak` §1
`\pagecounter` §2
`\pageno` §5
`\parenthborder#1` §3
`\perm#1:` §1
`\pftreadnote#1` §5
`\placeheaderbox` §2
`\placemultchoicebox` §2
`\placetotlasbox` §2
`\picture#1by#2(#3)` §5
`\probcounter` §5
`\problem` *Introduction*
`\problemlabel` §5
`\problemnameskip` §2
`\Professor#1` §2

`\scaledpicture#1by#2(#3scaled#4)` §5
`\setglobalperms#1` §1
`\setlabels#1#2#3#4#5` §3
`\setpermlist#1`
`\setproblempermutations#1`
`\sevenrm` §5
`\skipforfootlineofanswersheet` §2
`\smallfrac#1#2` §5

`\Tf` *Introduction*
`\tF` *Introduction*
`\threeone` §6
`\threetwo` §6

Pseudo-Index

`\title#1` §2
`\totals` §2
`\twocolumn#1` §2
`\twolinetitle#1#2` §2
`\twoone` §6
`\twotwo` §6

`\version` §5

`\writeanswersheet#1` §2
`\wrong` *Introduction*