Appendix A

An Indexing Facility for TEX

Terry Winograd and Bill Paxton
July 17, 1980

We have created a set of TEX macros and INTERLISP programs that generate an alphabetical index in various standard book formats. The index terms are sprinkled into the text, using a macro. As a side effect of compiling the file to produce pages, TEX creates an index file designed to be read by an INTERLISP program. This program can merge any number of index files and produce an alphabetized formatted index that can then be compiled by TEX to produce the final index pages.

It is easiest to understand this by looking at examples and seeing how they come out. First we will go through an extended example, then describe the features, then finally give the code. The sample index produced is:

```
Artificial intelligence, 70, 76, 82, 91, 526,
    529

Constantinople, 12ff. See also Istanbul.
  alien rule of, 20ff.
  Arab invasion of, 19, 31n.
  Crusades and, 22ff., 57
  founding of, 14, 16ff.
  Golden Age of, 17ff.
  Greek takeover of, 28-29
  Ottoman conquest of, 29-31
  Persian attack on, 18-19
  Venetian sack of, 31

Darwin, Charles, 82
Darwin, Max, 82
Death. See Eggs, Life.

Eggs
  fried, 530. See also Death.
  scrambled (yuk), 526
Everything else, 70. See also Life.

Indexing, 70, 527-30
  cross, 76, 529
    strategies, 526
    typography, 527ff.
  strategies, 529
Istanbul, 63. See also Constantinople.

Life, 82-91
```

The output of the INTERLISP program is designed to provide a great deal of flexibility in formatting the final index. We have developed three different styles; if none of them suit you, there is a good chance you can produce something that will. In addition to the "entry-per-line" style shown above, we offer run-in (paragraph) style and a combined style in which subentries all begin a new line, preceded by an em dash, but sub-subentries are run-in. In general, the entry-per-line style is best if for a complex index, paragraph style is more economical in terms of space, and the combined style is a compromise between the other two. Here is the previous example using these alternative styles.

Paragraph style

Artificial intelligence, 70, 76, 82, 91, 526, 529

Constantinople, 12ff.: alien rule of, 20ff.; Arab invasion of, 19, 31n.; Crusades and, 22ff., 57; founding of, 14, 16ff.; Golden Age of, 17ff.; Greek takeover of, 28-29; Ottoman conquest of, 29-31; Persian attack on, 18-19; Venetian sack of, 31 *See also* Istanbul.

Darwin, Charles, 82
Darwin, Max, 82
Death. *See* Eggs, Life.

Eggs: fried, 530. (*See also* Death); scrambled (yuk), 526
Everything else, 70. *See also* Life.

Indexing, 70., 527-30: cross, 76, 529 (strategies, 526; typography, 527ff.); strategies, 529
Istanbul, 63. *See also* Constantinople.

Life, 82-91

Combined style

Artificial intelligence, 70, 76, 82, 91, 526, 529

Constantinople, 12ff. *See also* Istanbul.
—alien rule of, 20ff.
—Arab invasion of, 19, 31n.
—Crusades and, 22ff., 57
—founding of, 14, 16ff.
—Golden Age of, 17ff.
—Greek takeover of, 28-29
—Ottoman conquest of, 29-31
—Persian attack on, 18-19
—Venetian sack of, 31

Darwin, Charles, 82
Darwin, Max, 82
Death. *See* Eggs, Life.

Eggs
—fried, 530. *See also* Death.
—scrambled (yuk), 526
Everything else, 70. *See also* Life.

Indexing, 70., 527-30
—cross, 76, 529: strategies, 526; typography, 527ff.
—strategies, 529
Istanbul, 63. *See also* Constantinople.

Life, 82-91

In addition to these major style variations, you can modify the TEX macros to control smaller details as well. For example, you can easily change the manner in which primary references are indicated from boldface to something else, or you can change the formatting for cross references to put them all in parentheses. Comments in the TEX macros should help you to make the necesssary changes to get the style you want.

These examples are the result of merging two index files, one of which came from the source file on the following page, which was called TESTINDEX.TEX.

The macro '\<' does indexing. It takes two arguments, the first of which is a single character and the second of which is the entry. The character means:

.	ordinary reference to the page on which it appears
:	boldface form of . (for use in giving primary reference)
-	begin span reference (e.g. '23-47') on this page
=	boldface form of -
←	end span reference on this page
|	'ff' reference to this page (e.g., '56ff.') use this to indicate the page on which a long discussion begins
!	boldface form of |
,	'n' reference to this page (e.g., '78n.') use this with a page reference to a footnote
;	boldface form of ,
*	author reference (see details below) to this page
+	cross reference (see details below)

Each call to '\<' contains a sequence of terms, separated by semicolons and is terminated with a '>'. The exceptions are the author reference (which calls for precisely two terms--last and first names) and the cross reference (which has the sequence of terms followed by an '=' followed by the phrase to be used in the '(see ...)'). Calls to the index macro do not affect the regular TEX output file at all. This was done because the exact wording of the index term is often not identical to a sequence of characters appearing in the text. It was more uniform to treat the index and text as always distinct. Also, the system carries through whatever capitalization you use in the index terms. The example here uses what is more or less standard in publishing. In this example we have used the character macro feature (\chcode = 13) to allow the character '<' to stand for the sequence '\<'. If you want to use '<' normally, you can skip this and simply use '\<' for index items.

TEX SOURCE FILE "TESTINDEX.TEX"

```
\input <tex>basic
\input index
\def\setpage #1 {\par\vfill\eject\setcount0 #1\setcount7 #1}
\openIndex testindex

\setpage 12
Our saga begins in the ancient city of Constantinople. <|Constantinople><+Constantinople=Istanbul>
\setpage 14
It was founded long ago. <.Constantinople;founding of>
\setpage 16
There are many things to say about its founding. <|Constantinople;founding of>
\setpage 17
The Golden Age of the city lasted for several hundred years. <|Constantinople;Golden Age of>
\setpage 18
The beginning of the end for the city was the fierce Persian attack. <-Constantinople;Persian attack on>
\setpage 19
After the Persian attack, there was an Arab invasion. <←Constantinople;Persian attack
on><.Constantinople;Arab invasion of>
```

```
\setpage 20
This marked the beginning of a period of alien rule. <|Constantinople;alien rule of>
\setpage 22
The Crusades had a major impact on Constantinople. <|Constantinople;Crusades and>
\setpage 28
The Greeks took over the city. <-Constantinople;Greek takeover of>
\setpage 29
The Ottoman conquest of the city soon followed. <←Constantinople;Greek takeover of><-
Constantinople;Ottoman conquest of>
\setpage 31
The Venetians sacked the city. <←Constantinople;Ottoman conquest of><.Constantinople;Venetian
sack of> In a footnote, we compare this to the earlier Arab invasion. <,Constantinople;Arab invasion
of>
\setpage 57
Once again the Crusades reached the city. <.Constantinople;Crusades and>
\setpage 63
The city became known as Istanbul. <.Istanbul><+Istanbul=Constantinople>

\setpage 70
This is material on indexing <.Indexing> and artificial intelligence <.Artificial intelligence> and
everything else, <.Everything else> which is what life is all about <+Everything else=Life>
\setpage 76
This is the main reference to cross indexing <:Indexing;cross> files. It also mentions AI <.Artificial
intelligence> and another reference to cross indexing <.Indexing;cross> happens to fall on the same page
so should not appear redundantly. If it had happened to fall across the page break I would want both
pages to have references.
\setpage 82
I begin discussing life <-Life> on this page, quoting from Charles Darwin. <*Darwin;Charles> and at
times from his brother Max <*Darwin;Max> who did research in AI. <.Artificial intelligence>
\setpage 91
Here ends our discussion of life <←Life> and AI <.Artificial intelligence> and other matters. It contains a
duplicate crossreference having to do with life and everything else. <+Everything else=Life>
\setpage 94
It also has described death <+Death=Life> to some extent.

\setpage 526
We also want to discuss scrambled eggs, <.Eggs;scrambled (yuk)> AI <.Artificial intelligence> and cross
indexing strategies. <.Indexing;cross;strategies>

\setpage 527
Now I begin the main discussion of indexing <=Indexing> with several pages on the typography of
cross indexes. <|Indexing;cross;typography>
\setpage 529
Here I mention cross indexing <.Indexing;cross> again, along with some general strategies
<.Indexing;strategies> useful in doing indices. I also both begin and end a discussion of AI. <-Artificial
intelligence> <←Artificial intelligence> This could happen with a begin and end that weren't separated
very far and might end up on either the same or adjacent pages. When they fall on the same page we
want a simple reference, not a span.
\setpage 530
This is the end of indexing, <←Indexing> which is more complex than fried eggs. <.Eggs;fried> Some
poets of the absurd have argued that death is really just an ultimate form of eggs. <+Death=Eggs>
<+Eggs;fried=Death>

\par\vfill\eject\end

% NOTE: the calls to \setpage are not normally used. They are
% included here to create an output that has lots of pages from
% a short test file and to include high page numbers. The system ordinarily
% works with the standard page breaking, indexing things by the page on
```

% which they appear.

When TEX compiles TESTINDEX.TEX, it produces the regular output TESTINDEX.PRESS, plus a file TESTINDEX.INDEX which contains:

```
<12;N;F;Constantinople>
<Istanbul;N;C;Constantinople>
<14;N;P;Constantinople;founding of>
<16;N;F;Constantinople;founding of>
<17;N;F;Constantinople;Golden Age of>
<18;N;S;Constantinople;Persian attack on>
<19;N;E;Constantinople;Persian attack on>
<19;N;P;Constantinople;Arab invasion of>
<20;N;F;Constantinople;alien rule of>
<22;N;F;Constantinople;Crusades and>
<28;N;S;Constantinople;Greek takeover of>
<29;N;E;Constantinople;Greek takeover of>
<29;N;S;Constantinople;Ottoman conquest of>
<31;N;E;Constantinople;Ottoman conquest of>
<31;N;P;Constantinople;Venetian sack of>
<31;N;N;Constantinople;Arab invasion of>
<57;N;P;Constantinople;Crusades and>
<63;N;P;Istanbul>
<Constantinople;N;C;Istanbul>
<70;N;P;Indexing>
<70;N;P;Artificial intelligence>
<70;N;P;Everything else>
<Life;N;C;Everything else>
<76;B;P;Indexing;cross>
<76;N;P;Artificial intelligence>
<76;N;P;Indexing;cross>
<82;N;S;Life>
<82;N;P;Darwin, Charles>
<82;N;P;Darwin, Max>
<82;N;P;Artificial intelligence>
<91;N;E;Life>
<91;N;P;Artificial intelligence>
<Life;N;C;Everything else>
<Life;N;C;Death>
<526;N;P;Eggs;scrambled (yuk)>
<526;N;P;Artificial intelligence>
<526;N;P;Indexing;cross;strategies>
<527;B;S;Indexing>
<527;N;F;Indexing;cross;typography>
<529;N;P;Indexing;cross>
<529;N;P;Indexing;strategies>
<529;N;S;Artificial intelligence>
<529;N;E;Artificial intelligence>
<530;N;E;Indexing>
<530;N;P;Eggs;fried>
<Eggs;N;C;Death>
<Death;N;C;Eggs;fried>
```

This file is then fed to LISP. The transcript in doing this was:

```
3←LOAD(TEXINDEX.COM]
compiled on  10-JUNE-80 11:57:41
FILE CREATED  10-JUNE-80 11:56:41
```

```
TEXINDEXCONS
<PAXTON>TEXINDEX.COM;20
4←INITIALIZE]
NIL
5←READINDEX(TESTINDEX]
NIL
6←WRITEINDEX(TESTOUT]
NIL
```

The program has three user-accessible functions: INITIALIZE(), READINDEX(NAME) and WRITEINDEX(NAME). INITIALIZE is called once on starting it up, READINDEX can then be called any number of times, reading in (and merging) .INDEX files. Finally WRITEINDEX is called once to write a file with extension .TEX which can then be compiled by TEX to produce the index. The resulting file, TESTOUT.TEX contains:

```
\indexStart
\indexChar{A}
\indexEntry0{Artificial intelligence, 70, 76, 82, 91, 526, 529}{{}-{}}
\indexChar{C}
\indexEntry0{Constantinople, \indexFF 12.\pageNumDot}{{\indexAlso{Istanbul}}+{
\indexEntry1{alien rule of, \indexFF 20.\pageNumDot}{{}-{}}
\indexEntry1{Arab invasion of, 19, \indexN 31.\pageNumDot}{{}-{}}
\indexEntry1{Crusades and, \indexFF 22., 57}{{}-{}}
\indexEntry1{founding of, 14, \indexFF 16.\pageNumDot}{{}-{}}
\indexEntry1{Golden Age of, \indexFF 17.\pageNumDot}{{}-{}}
\indexEntry1{Greek takeover of, \indexSpan 28-29.}{{}-{}}
\indexEntry1{Ottoman conquest of, \indexSpan 29-31.}{{}-{}}
\indexEntry1{Persian attack on, \indexSpan 18-19.}{{}-{}}
\indexEntry1{Venetian sack of, 31}{{}-{}}}}
\indexChar{D}
\indexEntry0{Darwin, Charles, 82}{{}-{}}
\indexEntry0{Darwin, Max, 82}{{}-{}}
\indexEntry0{Death}{{\indexSee{Eggs, Life}}-{}}
\indexChar{E}
\indexEntry0{Eggs}{{}+{
\indexEntry1{fried, 530}{{\indexAlso{Death}}-{}}
\indexEntry1{scrambled (yuk), 526}{{}-{}}}}
\indexEntry0{Everything else, 70}{{\indexAlso{Life}}-{}}
\indexChar{I}
\indexEntry0{Indexing, 70, \mainEntry{\indexSpan 527-30.}}{{}+{
\indexEntry1{cross, \mainEntry{76}, 529}{{}+{
\indexEntry2{strategies, 526}{{}-{}}
\indexEntry2{typography, \indexFF 527.\pageNumDot}{{}-{}}}}
\indexEntry1{strategies, 529}{{}-{}}}}
\indexEntry0{Istanbul, 63}{{\indexAlso{Constantinople}}-{}}
\indexChar{L}
\indexEntry0{Life, \indexSpan 82-91.}{{}-{}}
\indexEnd
```

Finally, this is put back through TEX with the initialization to produce the index.

COMMENTARY

Some of the non-obvious features are:

Page references appear in numerical order, with a simple reference preceding a span that begins on the same page. You can safely put in multiple references or begin-end pairs without knowing whether they will end up on the same page or adjacent pages. If there are multiple references to the same page, only one will appear. It will be bold if any of them were. If a span begins and ends on the same page a simple page reference appears instead. Unmatched begins and ends will be noticed by the INTERLISP program as it runs. It also notices and prints a message on the console about the presence of two spans starting on the same page.

Inclusive page numbers are elided to produce spans such as 107-9, 119-22, 245-49, 800-802. (The rule is: omit from the second number the digit(s) representing hundreds, except when the first number ends in two zeros, in which case the second number is given in full.) If you want page numbers to be given in full rather than elided, provide a second argument of T to the WRITEINDEX function.

Upper and lower case do not influence the alphabetical ordering of terms.

Cross-references are combined into a single list, in alphabetical order, regardless of where they were in the text. The LISP output indicates if there were any references to pages or any subterms so the TEX macros can produce *see* in some cases and *see also* in others. If the same cross-reference is given more than once in the source file, it appears only once in the output.

Spans (e.g. '3-5') are boxed, so they will not be split across lines.

Nesting of subterms can go any number of levels (assuming the width assigned for the index can stand it, and the TEX argument stack doesn't overflow). The deepest example here is 'Indexing, cross, strategies'.

When an entry is too long for its line, it is continued on the next line, indented two steps in the entry-per-line styles (see 'Artificial intelligence').

Spaces can appear in a term (e.g. 'Cognitive science'), as can parentheses (e.g. 'scrambled (yuk)'. In fact all characters except semicolon, '<' and '>' are treated exactly as TEX would normally treat them (e.g. single carriage returns, tabs and sequences of spaces become spaces, characters with special syntax like '{' and '\' have different effects, etc.). Because the TEX \send command is used, some slightly funny things happen when the text is used. For example, a built in TEX command (like \char) will get carried through the whole process and eventually produce its effect on the final output. A defined macro (including those defined in BASIC.TEX) will get expanded before the entry is sent to the file. If you really need a semicolon, '<' or '>' (or something else TEX won't let you use as a normal character), you have to put a \char in the index term. This may do funny things to the ordering, but then characters like semicolon and '>' are funny to alphabetize anyway.

Before the first entry for a letter, there is a call on the \indexChar macro with the character as argument. In our example, this macro simply puts out some blank space; in a larger index you might want to redefine \indexChar to put out the character too.

PROGRAMS

The TEX macros for the first pass are:

```
\gdef\lessthan{<}
\gdef\angbr#1{<#1>}
\chcode'74←13     % This allows < to stand for \< and gives us
          % an alternative way to put a < into an output (including a send)
```

```
\def\<#1>{}% Ignores index terms when index is not being generated.

\gdef\openIndex#1  {\open1=#1.INDEX
\gdef\<##1##2>{\if .##1{\doIndex{N;P}{##2}}\else
{\if :##1{\doIndex{B;P}{##2}}\else
{\if -##1{\doIndex{N;S}{##2}}\else
{\if =##1{\doIndex{B;S}{##2}}\else
{\if ←##1{\doIndex{N;E}{##2}}\else
{\if |##1{\doIndex{N;F}{##2}}\else
{\if !##1{\doIndex{B;F}{##2}}\else
{\if ,##1{\doIndex{N;N}{##2}}\else
{\if ;##1{\doIndex{B;N}{##2}}\else
{\if *##1{\doAuthor##2>}\else
{\if +##1{\doCross##2>}\else
{\error}}}}}}}}}}}}}

\gdef\doAuthor  #1;#2>{\doIndex{N;P}{#1,  #2}}
\gdef\doIndex#1#2{\send1  {\angbr{\count7;#1;#2}}}
\gdef\doCross#1=#2>{\send1  {\angbr{#2;N;C;#1}}}
```

Index terms will be ignored before a call '\openIndex foo', which opens the file FOO.INDEX. If no call to \openIndex appears, all indexing stuff will be ignored. We expect this to be the normal state--the \openIndex line will be added only when an index is being generated. The macros assume that there will be a counter (this version uses \count7) which will contain the page number. WARNING!! at the time the index terms are sent, the \output routine has already been executed, so if your \output routine bumps the page counter at the end (as many do), everything will be one off. Rewrite it to bump at the beginning.

The output file produced by these macros is read by the INTERLISP program. That program is included in full at the end of this memo since it is only a few pages long and since (as everyone knows) INTERLISP code is self-documenting.

Here are the TEX macros to format the index in the various styles.

```
% after \indexEntry comes the following:
%  level{term and pages}{{crossrefs}<flag>{subterms}}
%            <flag>="+" if have subterms, ="-" if don't

% someone will \let\indexEntry= one of the following
            % \indexLine for entry-per-line style
            % \indexPar for paragraph style
            % \indexComb for combined style

\gdef\indexLine#1#2#3{\setcount9#1\advcount9 by 2\noindent\hangindent\count9wd9 after
1\hskip#1wd9\gdef\crossIndexDot{.}#2\indexTail#3}

\gdef\indexTail#1#2{#1\par}
            % this hack puts out the crossrefs, discards the <flag>, does \par
            % and leaves the subterms to be read next

\gdef\indexPar#1#2#3{\if0#1{\indexPMain{#2}#3}\else
{\indexPSub{#2}#3}}

\gdef\indexComb#1#2#3{\if0#1{\indexCTop{#2}{#3}}\else
{\if1#1{\indexPMain{---#2}#3}\else
{\indexPSub{#2}#3}}}

\gdef\indexCTop#1#2{\noindent\hangindent 1wd9 after
1\gdef\crossIndexDot{.}#1\indexTail#2}

%\indexPMain and \indexPSub
%           #1=term&pages, #2=crossrefs, #3=subterms flag, #4=subterms

\gdef\indexPMain#1#2#3#4{\noindent\hangindent 1wd9 after 1
\gdef\crossIndexDot{.}#1\if-#3{}\else{\gdef\firstSubEntry{T}: #4}#2\par}

\gdef\indexPSub#1#2#3#4{\if T\firstSubEntry{\gdef\firstSubEntry{F}}\else{;
}\gdef\crossIndexDot{.}#1\if-#3{}\else{\gdef\crossIndexDot{.}\gdef\firstSubEntry{T} \
(\!#4)\gdef\crossIndexDot{.}}\let\indexSee=\indSee\let\indexAlso=\indAlso#2}

\gdef\indexSpan#1-#2.{\hbox{#1--#2}}
            % this is for reference to a span of pages

\gdef\indexFF#1.{#1ff.}
            % this is for reference to a long span of pages
            % "ff" stands for "following folios"

\gdef\indexN#1.{#1n.}
            % this is for reference to footnote
            % "n" stands for "note"

\gdef\pageNumDot{\gdef\crossIndexDot{}}
            % this appears at end of pages for entry if they end with "."
            % change \crossIndexDot to null so don't get double "."

\gdef\mainEntry#1{{\bf #1}}

\gdef\indexSee#1{\crossIndexDot\ {\it See }#1.}
\gdef\indexAlso#1{\crossIndexDot\ {\it See also }#1.}
\gdef\indSee#1{ ({\it See }#1)\gdef\crossIndexDot{.}}
\gdef\indAlso#1{ ({\it See also }#1)\gdef\crossIndexDot{.}}

\gdef\indexChar#1{\vskip 12pt} % can be changed if you want to put in letter headings
            % parameter is capital letter for next section of index

\gdef\indexStart{} % change this to put in your own heading

\gdef\indexEnd{\par\vfill\eject} % needs to be more complex for multiple columns
```

Important Note: In addition to loading the previous macros, you must \let\indexEntry= either \indexLine, \indexPar, or \indexComb depending on which style of index you want. We have used a box and the "wd" parameter to make it easier to produce indentations that are multiples of a constant width. You must do \save9 before you generate the index (e.g., \save9\hbox{---} is the right thing to do for the combined style). We have also made use of \count9 for doing the arithmetic. If you use this counter (or \box9) for other purposes, you need to change these. The output can be put into multiple columns by modifying the \output routine as described in the TEX manual.

A file containing these macros can be found on SCORE <tex.distrib>indexer.tex.

INTERLISP PROGRAM

The following pages contain the complete program. Any functions or constructs used in them that is not defined here can be found in the INTERLISP manual (October 1978 version). A file containing this program can be found on SCORE <tex.distrib>indexer.lsp.

```
(RECORD INDEXENTRY (UTERM TERM PAGES CROSSES SUBS))
(RECORD INPUTENTRY (PAGE BOLD TYPE . TERMS))
(RECORD PAGEREF (PAGEREFTYPE BOLD PAGENUM))
(RECORD SPANREF (PAGEREFTYPE BOLD PAGENUM ENDNUM))

(DEFINEQ

(INITIALIZE [LAMBDA NIL (* sets up readtable and creates empty index)
 (SETQ INDEXREADTABLE (COPYREADTABLE T))
 (for X from 1 to 127 do (SETSYNTAX X (QUOTE OTHER)
                    INDEXREADTABLE))
 (SETSYNTAX (QUOTE <)(QUOTE LEFTPAREN) INDEXREADTABLE)
 (SETSYNTAX (QUOTE >) (QUOTE RIGHTPAREN) INDEXREADTABLE)
 (SETSYNTAX (QUOTE ;)
       [QUOTE (MACRO (LAMBDA (FL RDTBL) (RSTRING FL RDTBL]
       INDEXREADTABLE)
 (SETSYNTAX (QUOTE %
)
       (QUOTE SEPR) INDEXREADTABLE)
 (SETQ WHOLEINDEX NIL])

(SPANREF? [LAMBDA (X) (* distinguishes SPANREFs from PAGEREFs)
 (CDDDR X])

(SAMEREF? [LAMBDA (X Y) (* ignores BOLD and PAGEREFTYPE properties)
 (AND X Y (EQUAL (CDDR X) (CDDR Y])

(MAINENTRY? [LAMBDA (X)
 (COND
  ((EQUAL "B" (FETCH BOLD OF X)) T)
  (T NIL])

(CONVERTSPANTOPAGE [LAMBDA (X) (* deletes field for ENDNUM)
 (RPLACD (CDDR X) NIL])

(READINDEX [LAMBDA (FILENAME) (* reads one index file, merging it into index)
 (PROG (INDEXFILE INPUT ENTRY TERM UTERM)
   (SETQ INDEXFILE (OPENFILE (PACKFILENAME (QUOTE BODY) FILENAME
                    (QUOTE EXTENSION) (QUOTE INDEX))
              (QUOTE INPUT) (QUOTE OLD)))
   (WHENCLOSE INDEXFILE (QUOTE EOF) (FUNCTION FILEND))
   (while (SETQ INPUT (READ INDEXFILE INDEXREADTABLE))
     do (SETQ TERM (CAR (fetch TERMS of INPUT)))
       (SETQ UTERM TERM)
       (SETQ TERM (MKATOM TERM))
       [COND
        ([NOT (SETQ ENTRY (GETPROP TERM (QUOTE ENTRY]
         (SETQ ENTRY (create INDEXENTRY TERM ← TERM UTERM ←(U-CASE UTERM)))
         (PUTPROP TERM (QUOTE ENTRY) ENTRY)
         (SETQ WHOLEINDEX (CONS ENTRY WHOLEINDEX]
       (PUTINDEX ENTRY INPUT (CDR (fetch TERMS of INPUT])

(FILEND [LAMBDA (FILE) (* handles end of file on input)
 (CLOSEF FILE)
 (RETFROM (QUOTE READ])

(PUTINDEX [LAMBDA (ENTRY INPUT SUBTERMS) (* puts one entry into index, checking for unmatched ends)
 (COND
  ((NOT SUBTERMS)
   (SELECTQ (MKATOM (fetch TYPE of INPUT))
      [P (replace PAGES of ENTRY
         with (CONS (create PAGEREF BOLD ←(MAINENTRY? INPUT)
                 PAGEREFTYPE ← NIL PAGENUM ←(fetch PAGE of INPUT))
```

```
                    (fetch PAGES of ENTRY]
            [F (replace PAGES of ENTRY
                with (CONS (create PAGEREF BOLD ←(MAINENTRY? INPUT)
                             PAGEREFTYPE ←(QUOTE F) PAGENUM ←(fetch PAGE of INPUT))
                    (fetch PAGES of ENTRY]
            [N (replace PAGES of ENTRY
                with (CONS (create PAGEREF BOLD ←(MAINENTRY? INPUT)
                             PAGEREFTYPE ←(QUOTE N) PAGENUM ←(fetch PAGE of INPUT))
                    (fetch PAGES of ENTRY]
            [C (replace CROSSES of ENTRY with (CONS (fetch PAGE of INPUT)
                                    (fetch CROSSES of ENTRY]
            [S (replace PAGES of ENTRY
                with (CONS (create SPANREF BOLD ←(MAINENTRY? INPUT)
                             PAGEREFTYPE ← NIL PAGENUM ←(fetch PAGE of INPUT))
                    (fetch PAGES of ENTRY]
            (E (for P in (fetch PAGES of ENTRY) when (SPANREF? P)
                do [COND
                    ((fetch ENDNUM of P) (UNMATCHED INPUT))
                    ((EQ (fetch PAGENUM of P) (fetch PAGE of INPUT))
                     (CONVERTSPANTOPAGE P))
                    (T (replace ENDNUM of P with (fetch PAGE of INPUT]
                  (RETURN)
                finally (UNMATCHED INPUT)))
            (SHOULDNT)))
    (T (PROG (SUBENTRY TERM UTERM)
           (SETQ TERM (CAR SUBTERMS))
           (SETQ UTERM (U-CASE TERM))
           [COND
            ([NOT (SETQ SUBENTRY (SASSOC UTERM (fetch SUBS of ENTRY]
             (SETQ SUBENTRY (create INDEXENTRY TERM ← TERM UTERM ● UTERM))
             (replace SUBS of ENTRY with (CONS SUBENTRY (fetch SUBS of ENTRY]
           (PUTINDEX SUBENTRY INPUT (CDR SUBTERMS])

(WRITEINDEX [LAMBDA (FILENAME NOELISION)    (* writes entire index onto an output file)
                (* if NOELISION then dont elide second number in span)
  (bind (LASTCHAR NEXTCHAR (ACODE ←(CHCON1 (QUOTE A)))
          (ZCODE ←(CHCON1 (QUOTE Z)))
          (TEXFILE ←(OPENFILE (PACKFILENAME (QUOTE BODY) FILENAME
                                  (QUOTE EXTENSION) (QUOTE TEX))
                          (QUOTE OUTPUT) (QUOTE NEW)))
          (OLDLEN ←(LINELENGTH 1500)))
    first (printout TEXFILE "\indexStart") for ENTRY in (SORT WHOLEINDEX T)
    do (SETQ NEXTCHAR (CHCON1 (fetch UTERM of ENTRY)))
      (COND
       ((OR (EQ NEXTCHAR LASTCHAR) (LESSP NEXTCHAR ACODE)
          (LESSP ZCODE NEXTCHAR))
        NIL)
       (T (SETQ LASTCHAR NEXTCHAR)
          (TERPRI TEXFILE)
          (printout TEXFILE "\indexChar{" (CHARACTER NEXTCHAR) "}")))
      (WRITEENTRY 0 (fetch TERM of ENTRY) ENTRY)
    finally (TERPRI TEXFILE) (printout TEXFILE "\indexEnd" T)
          (CLOSEF TEXFILE) (LINELENGTH OLDLEN) (CLEANINDEX])

(CLEANINDEX [LAMBDA NIL (* after finish writing index)
  (for ENTRY in WHOLEINDEX do (REMPROP (fetch TERM of ENTRY) (QUOTE ENTRY))
    finally (SETQ WHOLEINDEX NIL])

(WRITEENTRY [LAMBDA (DEPTH TERM ENTRY)
    (* writes a top-level entry along with its subentries.
       suppresses duplicates and checks for unmatched begins)
  (PROG (REFJUSTPRINTED CROSSREFS NORFF)
      (TERPRI TEXFILE)
      (printout TEXFILE "\indexEntry" DEPTH "{" TERM)
      [COND
       ((fetch PAGES of ENTRY)
        (for REF in (SORT (fetch PAGES of ENTRY) (FUNCTION REFBEFORE))
          when (NOT (SAMEREF? REF REFJUSTPRINTED))
          do (printout TEXFILE "," (COND
                  ((fetch BOLD of REF) " \mainEntry{")
                  (T " ")))
           (SETQ NORFF NIL)
          (COND
           [(SPANREF? REF)
            (COND
             ((NOT (fetch ENDNUM of REF)) (UNMATCHED (LIST TERM REF)))
             (T (ELIDESPAN (fetch PAGENUM of REF) (fetch ENDNUM of REF]
           ((EQ (fetch PAGEREFTYPE of REF) (QUOTE F))
            (printout TEXFILE "\indexFF " (fetch PAGENUM of REF) ".")
            (SETQ NORFF T))
           ((EQ (fetch PAGEREFTYPE of REF) (QUOTE N))
            (printout TEXFILE "\indexN " (fetch PAGENUM of REF) ".")
            (SETQ NORFF T))
           ((NULL (fetch PAGEREFTYPE of REF))
            (printout TEXFILE (fetch PAGENUM of REF)))
           (T (SHOULDNT)))
          (COND
           ((fetch BOLD of REF) (printout TEXFILE "}")))
          (SETQ REFJUSTPRINTED REF))
       (COND
        (NORFF (printout TEXFILE "\pageNumDot"]
     (printout TEXFILE "}{(")
     (COND
      ((fetch CROSSES of ENTRY)
       (SETQ CROSSREFS (SORT (fetch CROSSES of ENTRY)))
       (printout TEXFILE "\index" (COND
              ((OR (fetch PAGES of ENTRY) (fetch SUBS of ENTRY)) "Also{")
              (T "See{"))
             (CAR CROSSREFS))
       (SETQ REFJUSTPRINTED (CAR CROSSREFS))
       (for REF in (CDR CROSSREFS) when (NEQ REF REFJUSTPRINTED)
         do (printout TEXFILE ", " REF) (SETQ REFJUSTPRINTED REF))
       (printout TEXFILE "}")))
     (printout TEXFILE "}")
     (COND
      ((NOT (fetch SUBS of ENTRY)) (printout TEXFILE "-{}"))
      (T (printout TEXFILE " + {")
        (for ENT in (SORT (fetch SUBS of ENTRY) T)
          do (WRITEENTRY (ADD1 DEPTH) (fetch TERM of ENT) ENT))
        (printout TEXFILE "}")))
     (printout TEXFILE "}"])

(ELIDESPAN [LAMBDA (X Y) (* print elided form of span)
 [COND
  ((OR NOELISION (ZEROP (REMAINDER X 100))
     (NEQ (QUOTIENT X 100) (QUOTIENT Y 100)))
   NIL)
  (T (SETQ Y (REMAINDER Y 100]
 (printout TEXFILE "\indexSpan " X "-" Y "."])

(REFBEFORE [LAMBDA (X Y)
   (* ordering function used to sort page references.
     single pages are before spans and bold before non-bold if page is the same)
 (COND
  [(EQ (fetch PAGENUM of X) (fetch PAGENUM of Y))
   (COND
    ((SPANREF? X)
     (COND
      ((SPANREF? Y)
       (printout T "TWO SPANS START TOGETHER" T X T Y))
      (T NIL)))
    ((SPANREF? Y) T)
    (T (fetch BOLD of X]
  (T (LESSP (fetch PAGENUM of X) (fetch PAGENUM of Y])

(UNMATCHED [LAMBDA (INPUT) (* prints error message on terminal)
 (printout T "UNMATCHED ENTRY" T INPUT])
)

STOP
```