a portable program for grade II braille translation

by w.r. gerhart    j.k. millen    j.e. sullivan
the MITRE corporation

1953
Subject: DOTSYS III: A Portable Program for Grade 2 Braille Translation

Author: W. Reid Gerhart, Dr. Jonathan K. Millen, Joseph E. Sullivan

Dept.: D-73

Date: 14 May 1971

Contract No.: SR 21761

Contract Sponsor: Sensory Aids Evaluation and Development Center, Massachusetts Institute of Technology

Project: 1420

Issued at: Bedford, Massachusetts

Department Approval: [Signature]

MITRE Project Approval: [Signature]

Page 1 of 150 Pages

This document has been approved for public release.
ABSTRACT

This document describes DOTSYS III, a table-driven COBOL program for the translation of English text into grade 2 (or grade 1) braille. While general acquaintance with computer programming and the subject of braille translation would be helpful in reading the document, no special knowledge in these areas is presupposed. The program's method of operation, together with detailed instructions on using the program, on modifying or extending the translation heuristics, as defined in the tables, and on transferring the program to a new computer environment are all presented.

W. R. Gerhart
Intelligence and Information Systems

J. K. Millen
Intelligence and Information Systems

J. E. Sullivan
Intelligence and Information Systems

WRG:JKM:JES/ces
This document is intended to serve the needs of many different levels of interest. Those interested only in what the program will do, for example, need read only the introduction. Persons with a general interest in the subject of braille translation should add the section on method. Those who actually wish to use the program -- transcribers, editors, and keypunchers -- will, of course, want to read the appropriate parts of the Usage Section and may or may not be interested in method. A systems programmer or other person whose main interest is to get the program running in a new environment ought to read the "Transfer" Section. Finally, a programmer who may wish to modify the program should read the whole document, including the final section on maintenance.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LIST OF ILLUSTRATIONS</td>
<td>vii</td>
</tr>
<tr>
<td></td>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>I</td>
<td>SECTION I INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>SECTION II METHOD</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GENERAL</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>THE PRIMARY INPUT PROCESSOR</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>THE TRANSLATOR</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>The Buffer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>The Alphabet Table Search</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>The Contraction Table Search</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>The Buffer Shift</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Braille Sign Output</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>The State Transition</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>The Decision Table</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>THE STACKER</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>THE BRAILLE LINE COMPOSER</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>THE FINAL OUTPUT WRITER</td>
<td>8</td>
</tr>
<tr>
<td>III</td>
<td>SECTION III USAGE</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>INPUT</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Deck Setup</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>The Echo Option Card</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>The Tables</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>The Run Control Cards</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Text Input</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Notes to the Braille Editor</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>OUTPUT</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Echo Output</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Proof Output</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Elastomer Braille Output</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>RPQ Braille Output</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Punched Card Output</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Error Messages</td>
<td>31</td>
</tr>
<tr>
<td>IV</td>
<td>SECTION IV PROGRAM TRANSFER</td>
<td>34</td>
</tr>
<tr>
<td>TABLE OF CONTENTS (Concluded)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SECTION V</strong></td>
<td>PROGRAM MAINTENANCE</td>
<td>Page</td>
</tr>
<tr>
<td></td>
<td>WHEN MAINTENANCE MAY BE REQUIRED</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>TABLE-SIZE BOUNDS</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>REPLACING THE ALPHABET TABLE SEARCH</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>BY DIRECT INDEXING</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>CONTRACTION TABLE SEARCH ALGORITHM</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>General Rationale</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Notation</td>
<td>40</td>
</tr>
<tr>
<td><strong>APPENDIX I</strong></td>
<td>LISTING OF COMPLETE IBM 360 JOB DECK</td>
<td>43</td>
</tr>
<tr>
<td><strong>APPENDIX II</strong></td>
<td>PROOF OUTPUT FROM STANDARD SAMPLE</td>
<td>101</td>
</tr>
<tr>
<td><strong>APPENDIX III</strong></td>
<td>WORDS INCORRECTLY TRANSLATED IN 5808 PROBLEM WORDS</td>
<td>128</td>
</tr>
<tr>
<td><strong>APPENDIX IV</strong></td>
<td>DEFINITION OF STATE VARIABLES AND INPUT CLASSES</td>
<td>131</td>
</tr>
<tr>
<td><strong>APPENDIX V</strong></td>
<td>SUMMARY OF SPECIAL SYMBOLS</td>
<td>133</td>
</tr>
<tr>
<td><strong>APPENDIX VI</strong></td>
<td>EQUIVALENT SIGNS, SYMBOLS AND CODES</td>
<td>135</td>
</tr>
<tr>
<td><strong>APPENDIX VII</strong></td>
<td>TRANSLATOR - STACKER SIGN CODES</td>
<td>136</td>
</tr>
<tr>
<td><strong>APPENDIX VIII</strong></td>
<td>MISCELLANEOUS CODED VARIABLES</td>
<td>138</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>139</td>
</tr>
<tr>
<td>DISTRIBUTION LIST</td>
<td></td>
<td>141</td>
</tr>
</tbody>
</table>
LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Decision Table Schematic</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Sample of Proof Output</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>Elastomer Braille Output for Line in Figure 2</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>Punched Card Output for Line in Figure 2</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>Set-up Algorithm</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>Look-up Algorithm</td>
<td>42</td>
</tr>
</tbody>
</table>

LIST OF TABLES

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Alphabetical Contraction List Card Format</td>
<td>11</td>
</tr>
<tr>
<td>II</td>
<td>Alphabet Table Card Format</td>
<td>11</td>
</tr>
<tr>
<td>III</td>
<td>Contraction Table Card Format</td>
<td>13</td>
</tr>
<tr>
<td>IV</td>
<td>Right-Context Class Card Format</td>
<td>14</td>
</tr>
<tr>
<td>V</td>
<td>Decision Table Card Format</td>
<td>15</td>
</tr>
<tr>
<td>VI</td>
<td>Sign Table Card Format</td>
<td>15</td>
</tr>
<tr>
<td>VII</td>
<td>Suggested Number of Braille Lines Per Page</td>
<td>17</td>
</tr>
<tr>
<td>VIII</td>
<td>Variable Table Capacity References</td>
<td>37</td>
</tr>
</tbody>
</table>
SECTION I
INTRODUCTION

Braille, as it is used today in almost all contexts other than primer textbooks, is almost a system of shorthand and not simply a scheme for representing individual inkprint symbols in a tactile code. This system, called grade 2 braille, is defined in Reference 1. (A more direct transliteration is called grade 1 braille.) The chief device used to reduce the number of braille signs in grade 2 is the contraction. A contraction is the representation of an inkprint letter-group or whole word by a relatively short sequence of braille signs; for example, the word "receiving" is represented by the four braille signs for r, c, v, and g in succession. There are 189 letter-groups that may be contracted.

Unfortunately (at least from the standpoint of automating the translation process), a given contractable letter-group is not necessarily contracted wherever it appears. Moreover, the rules governing the use of contractions frequently involve such matters as pronunciation and meaning. For example, in the word "disease," the "dis" and the "ea" are normally contracted. However, the (now obsolete) meaning "lack of ease" is also defined for this word, and when "disease" is used with that meaning, the "ea" should not be contracted. This example, although admittedly farfetched, illustrates that in some circumstances even a human transcriber might have difficulty applying the rules. Cases routinely arise that are easy for a human transcriber, but still difficult for an essentially mechanical process — for example, distinguishing the musical note "do" from the verb "do." For these reasons, automatic natural-language to braille translation algorithms tend to be heuristic, which is to say fallible.

DOTSYS III is a computer program embodying such a natural-language-to-braille translation algorithm. As its name implies, it is an outgrowth of an earlier program, DOTSYS II, described in References 2 and 3.* The basic translation algorithm remains

* In order to make the present document as self-contained as possible, portions of References 2 and 3 have been incorporated with little or no change.
essentially the same, but a number of new features have been added, the translation quality has been improved, and better internal processing methods have been introduced. These improvements in capability have been offset by the improved methods, so that the overall speed of DOTSYS III remains about the same as DOTSYS II (1300 wpm on an IBM 360/50, 3330 wpm on a 360/65). DOTSYS II, in turn, owes the fundamentals of its translation algorithm both to previous work in the field of braille translation by computer and to elementary concepts in the theory of automata, logic design, and formal languages. The background references and relationships discussed in Reference 2 are relevant to DOTSYS III also.

The source language used in coding DOTSYS III is ASA Standard COBOL level 2 (Reference 4), with the COMPUTE verb the only feature used that is not level 1. (COMPUTE statements may easily be recast as level 1 statements if required.) This should enable DOTSYS III to be run on any computer having a COBOL compiler and a modest amount of core storage (approximately equivalent to 64,000 bytes of IBM 360 storage).

The input text is presented to DOTSYS III in the form of 80-character (punched card image) records. These can be manually keypunched directly from inkprint or produced by another program according to a fairly straightforward set of conventions. The output is the sequence of braille signs equivalent to the input text. Output can be produced in any of several forms, including "proof" output for a sighted editor and tactile (embossed) braille.

DOTSYS III is almost completely table-driven; i.e., details of the translation algorithm are determined by tables read in at execution time rather than by the program itself. DOTSYS III, as described in this document, comprises both the program and a standard set of tables. In principle, with modified tables, the DOTSYS III program would be capable of processing different kinds of text, such as text containing mathematical or technical notation, languages other than English, or text containing nonstandard symbols for format control.
SECTION II

METHOD

GENERAL

DOTSYS III comprises five cooperating processors: the primary input section, the translator, the stacker, the braille line composer, and the final output writer. It is the translator, as its name implies, that does most of the work of braille translation, and in fact, this section forms a sort of main loop or program, the others being in the form of subprograms called by the translator to do their work at the appropriate time. However, in order to follow the processing of a given piece of text from inkprint form to braille form, it is easiest to imagine the processors running sequentially, each one in turn operating on the output, or a collection of outputs, from the previous processor.

The following descriptions of these processors are idealized to some extent, so that the discussion does not become hopelessly mired in detail.

THE PRIMARY INPUT PROCESSOR

The primary input section reads the 80-characters (card-image) records. Columns 73-80 are discarded, so that the first character of a record logically follows character 72 from the previous record. This stream of characters is further collapsed by deleting all instances of the vertical bar character (|), and by deleting all consecutive blanks after the second in a series following a period (.) and all after the first in any other context. This collapsed stream of characters, one at a time, is the output of this section.

In the "self-checking" mode (described in detail in a later section), the primary input processor prepares the correct translation words for later comparison against the translator output.
THE TRANSLATOR*

The Buffer

The translation section operates on the stream of characters issued by primary input. As a first step, these are collected into a ten-character sliding window, called the "buffer," so that a group of characters can be examined.

The Alphabet Table Search

The leftmost character in the buffer is looked up in the alphabet table. In this table, there is exactly one entry for each symbol that may appear in the text, containing, among other things: (a) a code denoting the braille sign for that symbol, (b) the symbol's "input class," and (c) an index to that portion of the contraction table which pertains to this initial letter. An input class is an arbitrary numerical code with three distinct purposes, as will be seen.

The Contraction Table Search

The next step is to search that section of the contraction table indicated for this initial letter. In principle, this search may be visualized as a simple top-to-bottom entry-by-entry sequential search, although in fact a much faster tree-search algorithm (described in detail under "Maintenance") is used. An entry in the contraction table consists of: (a) a string of up to nine characters (ten, counting the implied initial letter); (b) a "right-context class" designator; (c) an input class code; (d) a shift count; and (e) a set of up to four braille sign codes.

For a match to occur on a particular entry, three conditions must be satisfied. First, the entire string for that entry must correspond to the buffer, or left substring thereof. Secondly, the input class for the entry determines a set of "state variable" conditions that must be satisfied. A state variable is a logical switch, having a

* The translator operates essentially as described in Section III of Reference 2, except that (1) the contraction table search has been speeded up considerably by using a modified binary search, which affects the table ordering rules slightly, (2) the STRING field delimiter in the contraction table has been changed from "$" to "|" (vertical bar), and (3) the decision table permits a new decision symbol "F" meaning "unconditional no."
value "yes" or "no" according to its meaning and the text already translated. For example, a state variable whose meaning is "after a digit" would have the value "yes" if the character immediately preceding the one leftmost in the buffer had been one of the digits 0-9 and would have the value "no" in all other circumstances, including initially. "Meaning" is, strictly speaking, defined completely by entries in another table which governs the setting of these switches, called the transition table. This table will be discussed presently. The mechanism by which the input class selects a set of state variable conditions to be tested is called the decision table; this table is also discussed in more detail in a separate section.

The third condition for a contraction table match to occur is that the right-context class be correct. If the right-context designator for the entry is blank, no right-context condition is imposed. Otherwise, the character immediately to the right of the matched string in the buffer is looked up in the alphabet table to determine its input class. Another table, called the right-context table is consulted to determine whether that input class is one of those listed as acceptable for this designator—e.g., the designator "P" (for "punctuation") may apply to input classes 2, 3, 13, and 14.

The contraction table search stops when a match occurs or the end of the table section for that particular initial letter is reached. In the latter event the shift count is taken to be 1, the input class and braille sign code revert to those found for the initial letter, and processing proceeds.

The Buffer Shift

The buffer is then "shifted" left by the amount of the shift count, with new characters from the input stream entering on the right.

Braille Sign Output

The braille sign code(s) are sent to the stacker section, constituting the output of the translator. These may directly represent braille signs, or they may be special control codes as is discussed in the section describing the stacker.

The State Transition

Finally, the input class is used to determine a set of transitions to be applied to the state variables. For each variable, the transition may be specified as "no change", "toggle" (change "yes" to "no" and "no" to "yes"), "set to yes" or "set to no."
After the state variable transition, the process is repeated, beginning with the alphabet table search.

The Decision Table

The decision table determines whether the current settings of the state variables permit the use of a given contraction table entry. It is best represented by a tableau in two parts, as depicted in Figure 1. The upper, or decision portion, has a row for each input class; the lower, or condition portion has one row per state variable. The number of columns is the same for both sections, and this number will depend upon the number of independent tests that may have to be made. The elements of the array are single characters, as given in Figure 1.

Processing of the table for a particular input class consists of a left-to-right scan of the associated row in the upper portion. If a "G" ("Go") symbol is encountered, processing stops with a "yes" decision. If an "F" ("Fail") is encountered, processing is likewise terminated, but with a "no" decision.

If one of the symbols "Y" or "N" is reached, then the corresponding column in the lower portion is compared against the current settings of the state variables. A "Y" in the \textit{i}th row implies that the \textit{i}th state variable must have the value "yes"; an "N" demands the value "no" and a dash is satisfied by either setting. If the specified conditions are satisfied for all the state variables, then processing is terminated with a "yes" decision if the upper portion had a "Y" symbol, and "no" if it was an "N". If one or more state variables does not have the value specified, then scanning of the decision section row is resumed.

When a dash is reached in the scan, the scan simply continues with the next column. If the scan encounters the end of the row without otherwise reaching a decision, then the decision becomes "no."

THE STACKER

The stacker operates upon the braille sign codes issued by the translator. In the simplest case, these codes are merely accumulated until the code for the blank braille sign is received—i.e., a braille word is finished. This word, constituting one entry in a stack, is normally then removed from the stack and issued as output to the braille line composer. In exceptional circumstances, two or more words may be held in a stack before release. This may be because the order
Figure 1. Decision Table Schematic
may have to be changed (in braille, units of measure are placed before the associated numeric quantity), or because the overall length of several words must be computed before issuing the first one—as when centering headings.

All special operations by the stacker, including delayed release and interchange of order, are directed by control codes issued by the translator. These are distinguishable from ordinary braille sign codes in that they have a value greater than 64. (See Appendix VII.)

The stacker maintains up to three distinct stacks, sharing a common area by borrowing entry fields from a linked free storage list. One stack is used for normal output; a second stack is used to collect the words in a heading (such as a chapter title); the third is used to hold the title (running head) if it is present.

In the "self-checking" mode (described in a later section), it is the stacker that compares each braille word against the correct translation prepared by primary input.

THE BRAILLE LINE COMPOSER

The line composer operates on braille words (stack entries) produced by the stacker. In each case, the length of the word will determine whether it can fit on the current braille line, an internal output buffer. If so, it is simply added thereto. Otherwise, the current line is sent to the final output writer, cleared, and started anew with the current braille word.

The line composer also concerns itself with counting lines to determine a new page condition, page numbering and titling, and similar matters.

THE FINAL OUTPUT WRITER

The output writer is actually a collection of processors, one for each distinct mode of output. For each mode selected, the associated processor produces actual output corresponding to the current braille line set up by the line composer.
SECTION III

USAGE

INPUT

Deck Setup

The complete input deck read by DOTSYS III consists of four main sections, in order:

(1) the echo option card;
(2) the tables;
(3) the run control cards; and
(4) the text.

Sections 1-3 must be sequence-numbered (in columns 73-80), in strictly increasing order.

The Echo Option Card

The echo option card determines whether a literal listing of the tables and run control cards will be printed on the SYSPRINT (normal printer) output device. The echo card itself is always so printed; printing of the text is controlled by one of the run control cards.

If column 1 of the echo card contains an "E," (for "echo") the listing is produced; if it contains an "N," (for "no echo") it is not. Columns 2-80 are ignored. For production use, the "N" option would be usual.

The Tables

Table Data in General

The tables are supplied with the DOTSYS III program and form an integral part of the process described by this document. However, circumstances may arise such that the user may wish to alter or augment the tables. One such circumstance might be a word found to be incorrectly translated by DOTSYS III, that occurs so often in a given text that it is impractical to force the correct translation by
special treatment (see the "$/$", "/_" and "_/" control symbols under "Forcing and Preventing Contractions" in "Text Input", below). In such a case, an addition to the contraction table is called for.

The tables and associated dimensioning cards are to be arranged in the following order:

1. the alphabetical contraction list;
2. the alphabet table;
3. the contraction table;
4. the card specifying the number of right-context classes;
5. the right-context table;
6. the card specifying the number of state variables;
7. the card specifying the number of input classes;
8. the transition table;
9. the card specifying the number of decision table columns;
10. the decision table; and
11. the sign table.

The formats of the tables and cards listed above follow. All two-column numerical fields must contain two digits; in particular, a number less than 10 must be given a leading zero when used in a two-column field. Numerical limitations stated in the form: "this number may not exceed ..." are appropriate for the table sizes used in the version of the DOTSYS III program listed in Appendix I. Refer to "TABLE SIZE BOUNDS" under "PROGRAM MAINTENANCE" (Section V) if changes in table size are contemplated.

The Alphabetical Contraction List

This table contains exactly 189 cards, one for each contraction defined in grade 2 English braille. The order of input must be alphabetical. The table is used in support of the self-checking feature (described under "Text Input" below). The format is as given in Table I.
Table I
Alphabetical Contraction List Card Format

<table>
<thead>
<tr>
<th>Columns</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>The inkprint contractable letter sequence, left-justified.</td>
</tr>
<tr>
<td>24-31</td>
<td>Four 2-digit braille sign codes (Cf. Appendix VI) representing the equivalent braille sign(s). 99's are used for filler on the right.</td>
</tr>
<tr>
<td>32-33</td>
<td>Presently ignored, but reserved for a fifth sign code.</td>
</tr>
</tbody>
</table>

The Alphabet Table

The alphabet table identifies all legal text input characters. The order of input is arbitrary, but for the sake of efficiency should normally be by decreasing frequency of occurrence of the symbol. (Note also that the order is related to that of the contraction table, q.v.) A dummy entry, with 99 in columns 18-19, should follow the last actual alphabet table entry. Including this card, the total number may not exceed 64. The card format is given in Table II.

Table II
Alphabet Table Card Format

<table>
<thead>
<tr>
<th>Columns</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The symbol being defined.</td>
</tr>
<tr>
<td>18-19</td>
<td>The input class (must be in the range from 01 to the number of input classes). Note: A card with 99 in this field signals that the end of the alphabet table has been reached.</td>
</tr>
<tr>
<td>21-22</td>
<td>The sign code to use when the symbol occurs in a computer-braille string (see &quot;Text Input,&quot; below and Appendix VI).</td>
</tr>
</tbody>
</table>
Columns | Contents
---|---
24-25 | The sign code to use in normal context (Cf. Appendix VI).
30-31 | 01 if the symbol may be used as one of a pair of symbols bracketing a letter denoting itself; 00 otherwise.

The Contraction Table

The contraction table contains not only contractions but many other sequences related to the heuristics of the translation process, and the definition of special control symbols.

Entries in the contraction table beginning with the same symbol must be grouped together, and these groups must be arranged in the same order as the corresponding symbols in the alphabet table. Also, symbols in the alphabet table that have no corresponding section in the contraction table are grouped, in any order, at the end of the alphabet table.

Within a section of the contraction table determined by a common initial letter, all entries having a given second character must also be grouped together. The order of input of these subsections is completely arbitrary and usually will vary from section to section as a function of conditional frequency. This grouping scheme is continued right up to the 10th character. In general, if two entries agree through the nth character, then all entries between them must also agree with them through the nth character.

A dummy entry, with 99 in columns 18-19, should follow the last actual entry. The total number of cards, including the dummy, cannot exceed 1,000.

Table III gives details of the contraction table card layout.
Table III

Contraction Table Card Format

<table>
<thead>
<tr>
<th>Columns</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>Character sequence, left-justified. If the string is shorter than 10 characters, then a vertical bar (</td>
</tr>
<tr>
<td>16</td>
<td>Right-context class designation may be specified (non-blank) only if the string is shorter than 10 characters.</td>
</tr>
<tr>
<td>18-19</td>
<td>Input class. Note: A card with 99 in this field signifies that the end of the table has been reached.</td>
</tr>
<tr>
<td>21-22</td>
<td>Shift count.</td>
</tr>
<tr>
<td>24-31</td>
<td>Four two-digit sign codes (Cf. Appendix VI). 99's are used for filler on the right.</td>
</tr>
</tbody>
</table>

The Card Specifying the Number of Right-Context Classes

As its name implies, this card contains the number of right-context classes that are to be defined. This figure is punched in columns 18-19. It cannot exceed 02.

The Right-Context Table

This table contains one card per right-context class; i.e., there are as many cards as signified on "the Number of Right-Context Classes" card, just previously described. The layout is given in Table IV.
Table IV
Right-Context Class Card Format

Columns | Contents
--- | ---
16 | A single-character designator for the class being defined—e.g., "P" for "punctuation."
24-31 | Up to four input class codes. All symbols having one of the listed input classes are implied to have the right context class being defined. If there are fewer than four input class codes, the last one is simply repeated to make four.

The Card Specifying the Number of State Variables

The number of state variables is punched in columns 18-19. This number may not exceed 10.

The Card Specifying the Number of Input Classes

The number of input classes is punched in columns 18-19. This number may not exceed 25.

The Transition Table

This table contains one card per state variable. On each such card, the \(i^{th}\) column contains a character signifying how the state variable is to be set when input class \(i\) is processed. An "R" signifies "reset" (set to "no"), an "S" means "set" (to "yes"), a dash (–) means "leave," and "T" means "toggle" (change to opposite state).

The Card Specifying the Number of Decision Table Columns

The number of columns in the decision table is punched in columns 18-19. This number cannot exceed 15.

The Decision Table

This table contains one card per decision table column. The lay-out of each card is given in Table V.
Table V

Decision Table Card Format

<table>
<thead>
<tr>
<th>Columns</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - nic*</td>
<td>The *ith card column contains the upper (decision) section symbol for the input class *i. It must be G, F, Y, N or -. (See &quot;The Decision Table.&quot;)</td>
</tr>
<tr>
<td>nic - (nic+nsv)*</td>
<td>The (nic + j)th card column contains the symbol denoting the condition imposed on the jth state variable. It must be Y, N or -.</td>
</tr>
</tbody>
</table>

The Sign Table

The sign table contains exactly 64 cards, one for each "ordinary" braille sign code. (Codes 00 and 64 both refer to sign 64.) The *i*th card defines code *i*. The layout is given in Table VI.

Table VI

Sign Table Card Format

<table>
<thead>
<tr>
<th>Columns</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-6</td>
<td>Dots, keypunched as periods, corresponding to the braille dots embossed for this character. The correspondence is:</td>
</tr>
<tr>
<td>Card Column</td>
<td>Braille Dot No.</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7-9</td>
<td>Three characters to be printed on proof output, denoting the most common meaning for this sign.</td>
</tr>
</tbody>
</table>

* *nic is the number of input classes, nsv the number of state variables.*
The Run Control Cards

The run control cards select options that remain in effect throughout the translation of the text--i.e., for the entire "run," or job step.*

There are 7 run control cards. The first four select the output modes (described under "output," below); any combination is permitted. On these, only column 1 is read; it should contain an "N" if the output mode is not wanted or a letter associated with the mode otherwise. Specifically, the options cards are, in order:

1. Proof output (P or N in column 1).
2. "Elastomer" braille (B or N in column 1).
3. "RPQ" braille (R or N in column 1).
4. Punched-card output (P or N in column 1).

The fifth and sixth control cards select the braille page dimensions. In both cases the required number is punched in columns 18-19:

5. The number of braille signs per line (not less than 2 nor greater than 40).
6. The number of lines per 11-inch page. Table VII gives suggested values for this item.

---

* On the IBM 360 under OS, it is possible to use IBM Operating System Job Control Language to run part of the text input with one set of DOTSYS III control cards, and change control cards for a succeeding part of the input. This could be done simply by using the IBM utility IEBGENER to copy the tabular input onto a temporary disk data set, say &TABLES. Then, instead of executing the catalogued procedure COBFLG just once, do it once for each part of the text input. The SYSIN data set to use is the concatenation of &TABLES with a DD * data set consisting of the control cards and the text of that part. Quite possibly, similar facilities exist on other systems.
Table VII
Suggested Number of Braille Lines Per Page

<table>
<thead>
<tr>
<th>Line Printer Setting</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines/Inch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proof Output</td>
<td>10</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>Elastomer Braille Output</td>
<td>15</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

The last card determines whether or not automatic titling and page numbering is to occur:

(7) If the top line of each braille page is to be reserved for a running title and automatically produced page number, a "P" should be punched in column 1 and the starting page number should be punched in columns 32-35. Otherwise, a "N" should be punched in column 1. In this latter case, no automatic page numbering is done and running titles, while still permitted in the input text, will not appear on the output.

Text Input

General Keypunching Rules

Text is punched in columns 1-72 of each text input card using an EBCDIC keypunch (e.g., IBM 029) or the equivalent multiple punches on another model. Letters, numbers, and punctuation are reproduced as they appear in normal typewritten English text, with the exception of quotation marks, accent marks, mathematical symbols, and brackets ([,]). However, additional symbols must be added to the text to indicate capitalization, italics, and format controls, to force or prevent improper translations in exceptional cases, and to define the correct translation for self-checking purposes.

Column 72 of a card is considered to be adjacent to column 1 of the next card. In general, any number of spaces will be collapsed automatically into a single space (refer to the description of the
primary input processor), except following a period, where a number of spaces greater than or equal to 2 will be interpreted as 2 spaces; (e.g.:

THE START. THE END.

is interpreted as

THE START. THE END.)

**Capitalization**

If the first letter of the inkprint word is capitalized, the key-punched word must be preceded immediately by a logical-not sign (¬). If the whole inkprint word is capitalized, the key-punched word must be preceded immediately by two logical-not signs (¬¬). When Roman numerals are written as capital letters, a single logical-not sign must be used before a single letter and two logical-not signs must be used before numerals containing two or more letters.

**Italics**

If only one, two, or three successive inkprint words are italicized, each of the words must be preceded immediately by an underline( _) when keypunched.

If four or more successive inkprint words are italicized, the first word must be preceded immediately by two underlines ( _ _) and the last by one underline.

**Indicating Contractions in Self-Checking Mode**

If the self-checking feature of DOTSYS III is to be used, then all contractions which should be made must be surrounded by vertical bars ( | ). Refer to the section entitled "Self-Checking", below.

**Ordering**

When two or more special signs must be punched (e.g., an italicized capital letter), the ordering should be:

- Italic sign (_)
- Capital sign(s) ( ¬ or ¬¬ )
- Accent sign (¢)
- Vertical bar (|)
Forcing and Preventing Contractions

The form of any contractable letter group can be forced to occur, regardless of context, by surrounding the letter group with the symbols "/_" and "/_". For example:

\[a/_dd_/\]

would cause the "dd" contraction to be used even though otherwise the program would not (and should not) use it at the end of a word.

A contraction that would otherwise occur can be prevented by introducing the null replacement symbol $/ (see below). For example,

\[\text{DISE}$/\text{ASE}\]

will prevent the "EA" contraction.

Replacement Symbols for Special Characters

Mathematical Symbols. Symbols such as + (plus), - (minus), = (equal"), > (greater than), and < (less than) must be spelled out as words, even though they appear on the keypunch.

Quotation Marks. The single quote character on the keypunch (') is always taken to mean an apostrophe; thus, special symbols must be used for single quotes.

$' is punched for a left single quote.

$'R is punched for a right single quote.

Ordinarily, the double quote (" on the keypunch may be used for both left and right double quotes. However, double quotes within the scope of another pair of double quotes must be represented by special symbols.

$" is punched for a left double quote within quoted text.

$"R is punched for a right double quote within quoted text.

Accent Marks. Any accent or other diacritical mark used with a letter (such as é, è, ë, à, ç) is represented by preceding the keypunched letter with a cents sign (ç). For example, Abbe is keypunched →ABBcE.

* A keypunched equals sign is interpreted as a letter sign.
Brackets.

< is punched for a left bracket ([:].

> is punched for a right bracket (]).

Short Syllable Sign. To insert a short syllable sign, the special symbol $SV$ is used.

Long Syllable Sign. To insert a long syllable sign, the special symbol $LV$ is used.

End of Poetry Foot Sign. To insert an end of poetry foot sign, the special symbol $FT$ is used.

Caesura Sign. To insert a caesura sign, the special symbol $CS$ is used.

Null Symbol. The symbol $/$ is a null replacement symbol (not a blank) and is generally used to prevent contractions (see above).

Forced Blanks. To produce multiple blanks or otherwise control their placement, the symbol $B$ is provided. One blank is produced for each occurrence of the symbol (e.g., A$B$B$B$BC produces A   C).

Format and Mode Control Symbols

Keypunching Rule for Spaces Around Format Controls. Unlike replacement symbols where spaces before or after the symbol are interpreted as spaces (i.e., word dividers), any spaces before or after format symbols are ignored. However, the general rule is to provide spaces around each format symbol to avoid possible ambiguity (e.g., $LVOICE$ would be interpreted as $LV$ OICE even though $L VOICE$ may have been intended). Otherwise, the control symbols will usually operate properly regardless of the presence or absence of blanks.

Paragraphs. The symbol $P$ must be keypunched to indicate the beginning of a new paragraph. The text following the $P$ is started on the next line after two spaces (that is, starting in the third cell position).

New Line. The symbol $L$ may be used to begin a new output line. Caution: at most one line will be skipped, even if the $L$ symbol is repeated. To skip more than one line, see "SKIP MULTIPLE LINES."
Skip Multiple Lines. The symbol $SLnnb, where b is a blank and nn is a two digit number (with a leading zero, if necessary) will skip the number of lines indicated, and output will continue from the left margin.

New Page. The symbol $PG may be used to begin a new page.

One-Time Tabulation. If the symbol $TABnnb is punched, where n's represent digits and b represents a blank, the text beginning immediately after the blank will be started at column nn. Tabulation is implemented by the automatic insertion of spaces into the output and will therefore proceed to the next line if nn is less than or equal to the present column number. A leading zero must be supplied if the column number is 9 or less.

Permanent Tabs. Numbered tabs can be set so that the user can right, left, or decimal point justify a word or number on any column. For example, the symbol $STB4L03 means set tab 4 to left justify on column 3. The symbol $STB means to set tab, followed by the one digit number of the tab to be set, followed by an L for left justification (alignment of the left most character), R for right justification or a D for decimal justification. The last two digit numbers are the column on which justification is to take place. After a tab has been set, it may be executed any number of times by inserting the symbol $# followed by the one digit number of the tab to be executed. (Example: $STB4L03 $#4 LEFT—will left-justify the word LEFT on column 3. The symbol $#4 can be used any number of times.) If a tab is called in a cell position less than or equal to the present position, output will begin on the next line.

The definition of a given tab number may be changed as often as desirable in the text. Initially, all tabs are set to left-justify on column (5 x tab number).

Titles. Titles are placed between the symbols $TLS (Title START) and $TLE (Title END), (e.g., $TLS THIS IS A SAMPLE TITLE $TLE.) After a title has been inserted, each subsequent page has a centered title as its first line (the same line which contains the page number). Pagination must be turned on for titles to appear on output (see "The Run Control Cards"). If a new title is entered, it takes the place of the old on all future pages. Entering the title does not automatically turn to a new page.

Headings. These are similar to titles except that the control symbols are $HDS and $HDE and that headings are a one time occurrence. Headings are centered on the next line with subsequent input beginning in column 1 of the line after the heading. Headings that overflow begin in Column 4 of successive lines.
Poetry. The symbols $PTYS$ (Poetry START) and $PTYE$ (Poetry END) are placed before and after all poetry text input. In this mode, the continuations of all poetry lines which exceed the physical length of the output line will be indented two spaces.

Turning the Self-Checking Mode On or Off. The symbol $SCON$/$/$/$/$/$/$/ is used to turn self-checking on and $SCOFF$ is used to turn self-checking off. (See "Self-Checking" below.)

Self-Checking

Self-checking is a feature that allows DOTSYS III to be used to check itself (and the contraction table) against a text specially marked to indicate the correct translation. This is accomplished by automatically comparing the results of two translations:

(1) the normal process, ignoring the special markings, and

(2) a trivial special process, wherein the markings are used to determine where contractions occur.

Discrepancies are flagged on the output, so that the clerical effort required to check the program's correctness is greatly reduced.

A deck containing 5,808 typical and problem words and phrases, taken from the Transcribers' Guide to English Braille (Reference 6), has been punched with the special markings necessary for self-checking.

In order to turn on self-checking, a control symbol ($SCON$/$/$/$/$/$/) must be included in the text. The text following this symbol is punched normally except that vertical bars ($|$) should immediately surround any sequence of letters that should be contracted in a word. For example,

```
    ever y th/ing
```

should be punched
```
|EVER|Y|TH|ING|
```

Note: A vertical bar at the end of the word and followed by a blank may be omitted; i.e.,
```
|EVER|Y|TH|ING
```

is an acceptable variation.

* Throughout this report, an underlined sequence of two or more letters in an example denotes a group contractable in braille. Adjacent contractable letter groups are separated by a slash. This is consistent with common practice (Refs. 1 and 6).
Those braille words not passing the comparison test are output as they ordinarily would be but flagged by appending two braille "for" symbols (all dots). In the proof output, the comment "TSK nnn" appears to the right of any line containing a flagged word. (If the first word in a line is in error, the TSK appears on the preceding line. In this context, a word is any sequence of non-blank braille symbols.) The nnn is simply a sequence counter, incremented by one for each discrepancy.

There is only one known situation where, assuming correct input, the program will fail to flag a word that is incorrectly translated. That is when a letter sign should be added by the translator and is not; for example, the "ab" of

\[ ab \text{ _initio} \text{ (italicized)}, \]
punched

\[ _{\text{AB} \text{ _|IN|ITIO}}, \]

should be preceded by a letter sign in braille. These cases are indicated by an asterisk in the Transcribers' Guide, so that manual checking of these is fairly easy.

More commonly, words are flagged as wrong that are in fact correct. There are a number of situations where this can occur:

(1) Synchronization: When, for whatever reason, the correspondence between inkprint (punched) words (sequences of non-blanks) and braille words is not one-for-one, a possibly spurious error due to synchronization will be flagged. For example, the first space in the phrase "by the by" is removed in braille; the word by will not compare with the "word" by the, causing a synchronization error. Control inputs ($TAB, etc.) also cause spurious errors of this class, but controls are not normally used in problem word lists. Typically, the word where the synchronization failed and the one after it are flagged before proper synchronization is reestablished.

(2) "Perceiving": For the sake of saving space, only the first four braille sign codes are read into the alphabetical contraction table, even though five are punched on the table input card. This means that "perceiving," the only 5-sign contraction, is not properly represented in the list and so words containing this contraction will be improperly flagged.

(3) Periods: Words containing periods will be flagged because the alphabet table assumes that periods are decimal points unless the contraction table logic explicitly overrides this assumption.
All of the spurious error flagging (or failing to flag), except possibly for the synchronization errors, could be corrected with somewhat more logic and/or storage cost in the program. At least in the case of the 5,808 problem words, these problems have not been sufficiently bothersome to warrant the expenditure.

Octal Braille

Octal braille permits the user to generate arbitrary braille cells directly. A particular braille cell is selected by a two-digit code. This code is the sum of the numbers associated with the dots in the cell, according to the following association scheme:

<table>
<thead>
<tr>
<th>Braille Cell Dots</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 \ . \ . \ 1</td>
</tr>
<tr>
<td>20 \ . \ . \ 2</td>
</tr>
<tr>
<td>40 \ . \ . \ 4</td>
</tr>
</tbody>
</table>

The codes for the desired braille cells are placed in sequence, following the special sign $OCT$. The first blank after $OCT$ terminates the sequence.

The code is actually an octal representation of the braille cell. Example:

$OCT521163107000307270$

will translate into

```
• • • • • • • •
• • • • • • • •
O C T A L B R L
```

Computer Braille

Computer braille is similar to octal braille except that it is driven by characters rather than two-digit numbers and is initiated by the special sign $CPB$. When using computer braille, a character is represented by the two-digit braille sign code punched in columns 21-22 of the corresponding alphabet table input card. This two-digit number is determined by the sum of the numbers associated with each braille dot in the table:
Example:

Entry in alphabet table:

<table>
<thead>
<tr>
<th>Column 1</th>
<th>18</th>
<th>21</th>
<th>24</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>01</td>
<td>41</td>
<td>15</td>
<td>00</td>
</tr>
</tbody>
</table>

Text Input:

$CPB%%%%$

will cause output

... ... ...

... ...

Notes to the Braille Editor

Role of the Editor

This section indicates where the intervention of the editor is required and presents the tools available to the editor to implement his intervention. Generally speaking, the nineteen problem situations listed in the Memorandum on Braille translation (Reference 5) by R. L. Haynes summarize where the intervention of the braille editor is needed.

There are basically two ways in which the editor can ensure proper translation in problem situations: (1) instructing the keypunch operator to add certain symbols to the input text; and (2) modifying the tables which control DOTSYS III. Modification of tables is explained under "Tables" above; the only time it would be done under normal circumstances would be when a problem word occurs often in a particular body of text; in that case, its correct translation would be added to the contraction table.

In the remainder of this section we shall discuss the special symbols which can be inserted in the input text and the situations in which they are helpful or necessary. These symbols are the letter sign (=), the grade switch ($G$), the division symbol ($/$), the
forced-contraction symbols (/, _, /), octal braille sign ($OCT ...), and the termination symbol ($T). The editor's attention is also directed to the sections on computer braille and the self-checking feature.

**Letter Sign (=)**

The editor must insert the letter sign when:

1. a letter which means a letter stands alone and is not followed by a period indicating an abbreviation and is not italicized or surrounded by double quotes or parentheses.

2. the capitalized or uncapitalized letter "a", "i", or "o" requires a letter sign in the braille, except when used after a number or as a word.

3. combinations of letters that could be confused with short-form words appear in the input text. (It is suggested that a contraction table entry be used to insert the letter sign if such a letter combination occurs frequently. The tables already contain a number of frequently used abbreviations.)

In other situations the program inserts the letter sign automatically where necessary (where one has not already been inserted in the input).

**Grade Switch ($G$)**

An occurrence of the grade switch, $G$, changes the mode of translation from grade 2 to grade 1 or vice versa. It must precede and follow any sequence of characters in which no contractions are to be used, such as a foreign word.

**Division Symbol ($/$)**

The division symbol is the most useful and versatile tool of the braille editor, although its operation is simple: it merely prevents contraction of a letter group which crosses it. For example, for the lisped word "thentury", the "the" contraction is avoided by inserting the division symbol after the "th", thus: TH$/$ENTURY. It may be placed between the parts of compound words, such as NUT$/$HATCH. It may be placed between prefixes and stems, as in BI$/$NOMIAL, or indicate syllable division, as in PERITO$/$NEUM and SKI$/$DADDLE.

It must be used in a time interval after the hyphen separating minutes from hours, as in 9:30 - $/$10:30, in order to produce a number sign correctly.
The division symbol may also be used in cases where the symbols to be translated are coincidentally DOTSYS III control symbols. For example, $$/TAB22$$ will be translated as "$TAB22" literally, avoiding the control function "tab to column 22." Even "$/" may be generated for output by supplying ""$$/'As input.

**Forced-Contraction Symbols (/\_\_, /\_)**

These symbols are used to force a contractible letter group to be contracted. See "Text Input" for a complete description.

**Octal Braille Sign ($OCT ...)**

This facility allows an arbitrary sequence of braille signs to be inserted in the output. This is an alternative to $$/$$ (q.v.) when the symbols to be translated into braille happen to be DOTSYS III input control symbols, and the control function is not desired. Another use would be in the preparation of special tactile effects, such as drawing diagrams using braille dots. See "Text Input" for a complete description.

**Termination Symbol ($T)**

This translates into the double sign dot-6, dot-3, in the output where required (Reference 1, Rule II.11a).

**OUTPUT**

**Echo Output**

When selected by the echo card (q.v.), a literal listing of the table input and run control cards is produced following the image of the echo card itself, which is always printed. This printout is placed on the SYSPRINT logical device, preceding the proof output (q.v.), if any. Error messages (q.v.) may be interspersed with the echo output.

**Proof Output**

Though proof output is optional (see "Run Control Cards"), it is normally called for except, perhaps, in final production runs. It is essentially a printout for the use of a sighted braille editor; the braille signs are represented as printed dots. This printout occurs on the same logical device (SYSPRINT) as the echo output and error messages (q.v.); error messages may be interspersed with the normal proof output.
The first three pages of proof output display the contents of the right context table, the decision table, and the transition table in a form much easier to read than the literal echo listing.

The remainder of the output is primarily a page-by-page, line-by-line representation of the braille output, using periods for braille dots. Below the braille sign are up to three characters intended to help identify the sign. If the sign represents a letter, for example, the letter itself is printed below the sign. The identification characters depend only on the sign (as determined by the sign table, q.v.) and not its context. For example, the sign for "K" has the identification character "K" even when it represents the word "knowledge." Below the identification characters is printed the braille sign code in the range 0 through 63, which may be used to check the punched output. Figure 2 contains a sample of proof output.

A literal listing of the text input cards, as read, is also produced between the lines of braille output (where a "braille output line" is actually a group of five printed lines). These card images will generally occur somewhat sporadically, with two or more occurring together at times and none between pairs of braille lines in other cases. (For separation, a blank line is printed in the latter case.) Typically, a given input word and the corresponding output are separated by about two braille lines.

If self-checking is in effect, the word TSK with a number immediately under it may appear on the far right of the page, to call attention to mistranslations. See the section entitled "Self-Checking."

Elastomer Braille Output

Elastomer braille output consists of the braille signs (dots) only, with each line produced in mirror-image. This output is suitable for a form of embossing on a line printer which has been modified by placing an elastic band between the print hammers and the paper. The elastic band might be a rubber band, soft polyurethane, a garter belt, or whatever the user's ingenuity can provide. It can be held on by tape or wire at the ends, or, on an IBM 1403 line printer, by hooks obtainable free of charge from IBM if requested under the number RPQ 818047. The number of lines per inch should be set as close as possible to ten.

This output, if selected, is placed on logical device SYSBRL. Figure 3 contains a sample of Elastomer braille as it would appear on the printed, or depressed dot (as opposed to raised dot) side.
Figure 2. Sample of Proof Output
Figure 3. Elastomer Braille Output for Line in Figure 2
RPQ Braille Output

RPQ braille is similar to elastomer braille (q.v.) in most respects, with an additional modification to the print chain (IBM RPQ F19229) so that spacing of the dots is closer to the braille standard. This output, if selected, is placed on logical device SYSRPQ.

Punched Card Output

Punched card output makes possible the processing of DOTSYS III's braille output by other programs—for example, to produce tactile braille on a computer system which can drive an embosser but cannot support a suitably modified DOTSYS III.

Each card represents one braille line; page boundaries are not represented. Forty two-digit braille sign codes are punched on each card, with 00's filling out the line on the right. Figure 4 illustrates part of a card containing punched output.

Punched output is placed on logical device SYSPUNCH. Of course, on systems supporting a logical file concept, this device need not be physically a card punch but could be, for instance, a tape containing equivalent card images.

Error Messages

Error messages are unconditionally printed on SYSPRINT, thereby appearing intermixed with echo and proof output, if any. The following is a list of the possible messages. For each entry, "A" is an explanation, "B" is the program's automatic action, and "C" is the suggested user response.

(1) NEW CHAR X

A. The character printed (X) has appeared in the input text but does not have an alphabet table entry.

B. The character is replaced by an asterisk and processing proceeds. (Note: processing stops if asterisk is not in the alphabet table.)

C. Correct the text input or provide an alphabet table entry.
Figure 4. Punched Card Output for Line in Figure 2
(2) **TABLE SEQUENCE ERROR**

A. A contraction table section is out-of-sequence with respect to the alphabet table order.

B. The "echo" option is turned on, if not on already, so that at least the remainder of the tables will be printed. Reading of the tables is continued but processing of the text is suppressed.

C. Refer to "Input" for a discussion of the relationship between the alphabet table arrangement and that of the contraction table, and rearrange accordingly.

(3) **FOLLOWING CARD(S) OUT OF SEQUENCE**

A. An out-of-order card was found in the section of the input containing the echo card, the tables and the run control cards.

B. Processing of this section is continued but processing of the text is suppressed.

C. Reestablish correct order or correct the sequence number.
SECTION IV
PROGRAM TRANSFER

This section contains a checklist of steps required to bring
DOTSYS III into operation on any computer having a Standard COBOL
compiler and sufficient core storage to execute DOTSYS III. 'Standard
COBOL' means COBOL according to Reference 4 having the level 1 nucleus,
table handling, and sequential access. The core storage required
depends on the compiler and on the size of the contraction table.
With the IBM 360 level F compiler and 1,000 contraction table entries,
about 59,000 bytes are required.

It is assumed here that the program and tables will be keypunched
from the listings appended to this manual. If a card deck or the
equivalent has been provided, information about how it was produced,
and how it is to be used, should accompany it.

The program can be made about fifteen per cent faster, without
affecting its translation capability, by replacing the alphabet
search by a machine-dependent indexing scheme, at the expense of
reducing further transferability and increasing the program size.
The method of doing this is explained under "Maintenance."

Here is the checklist of steps to transfer DOTSYS III:

(1) Rewrite the environment division of the program in
accordance with the COBOL manual for your computer and peripheral
units. Keep in mind that SYSINPUT is the input file, and SYSPRINT,
SYSPUNCH, SYSBRL, and SYSRPQ are output files. Other information
about them is in the file description (FD) entries in the data
division.

(2) Check the file description entries (in the file section of
the data division) to ensure that the block size and label records
are specified correctly for your installation.

Punched output records may or may not begin with a control char-
acter (for pocket selecting), and the control character, if present,
varies with the installation. DOTSYS III is set up to place a control
character in the beginning of each punched output record; the control
character used is the value of the data item POCKET-SELECT in the
working storage section. If no control character is to be used, remove the 02-level data item FILLER in CODED-OUTPUT, and change the sentence

WRITE CODED-OUTPUT AFTER POCKET-SELECT

in the BREAK6 paragraph of the output section to

WRITE CODED-OUTPUT

(3) Check the COBOL character set at your installation against the characters in the program. Some characters, such as the single quote ('), greater than (>) , less than (<), and equals (=), may have to be replaced by other characters or expressions.

(4) Check the character set of your computer and keypunch against the characters in the tables. Remove or replace table entries containing illegal characters. Inkprint characters having no single-character machine-readable form may be represented by multiple-character symbols, just as in the case of single quotation marks. They are implemented via contraction table entries. Be sure to inform the keypunch operator of your choices in this matter.

(5) If your COBOL does not permit the COMPUTE verb, replace each occurrence of it by an equivalent sequence of individual arithmetic operations. For example, the sentence "COMPUTE I = N * M + J." may be replaced by the two sentences

MULTIPLY N BY M GIVING I.

ADD J TO I.

(6) Compile the program to obtain an object deck. If the object program is too large, try leaving out the table display routine, from TABLE-DISPLAY through SKIP-DISPLAY. If the self-checking feature is not to be used, another possibility would be to shorten the alphabetic contraction table, or even to remove it and all associated code. As as last resort, decrease the contraction table bound (and shorten the table input accordingly).

(7) Execute the program, using standard test case input if possible.
SECTION V
PROGRAM MAINTENANCE

WHEN MAINTENANCE MAY BE REQUIRED

This section presents certain details of the COBOL implementation of DOTSYS III. It is for use in making changes in the tabular input, and in the program itself, that may be made necessary or desirable by the transfer of DOTSYS III to another installation, a change in peripheral units, a change in the braille translation rules, or a desire to improve the conformity of the translation with the braille ideal.

TABLE-SIZE BOUNDS

The maximum capacities for a number of tables are conceptually variable, in that only a few OCCURS clauses or other references need be changed and the program recompiled, to effect an increase or decrease. These items are listed in Table VIII.

Table VIII
Variable Table Capacity References

<table>
<thead>
<tr>
<th>Table</th>
<th>Program Card Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic Contraction</td>
<td>00033100, 00039800, 00091700</td>
</tr>
<tr>
<td>Alphabet</td>
<td>00024100</td>
</tr>
<tr>
<td>Contraction</td>
<td>00020800</td>
</tr>
<tr>
<td>Right Context</td>
<td>00025000</td>
</tr>
<tr>
<td>State Variables</td>
<td>00025900, 00026200</td>
</tr>
<tr>
<td>Input Classes</td>
<td>00025800, 00026300</td>
</tr>
<tr>
<td>Decision Table (width)</td>
<td>00025700</td>
</tr>
</tbody>
</table>
Table VIII (Concluded)

Variable Table Capacity References

<table>
<thead>
<tr>
<th>Table</th>
<th>Program Card Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stacks (no. of entries)</td>
<td>00022500, 00036000, 00036100</td>
</tr>
<tr>
<td>Stack (entry width)</td>
<td>00023000, 00101000, 00102600, 00133100</td>
</tr>
<tr>
<td>Self-Checking Ring (entry width)</td>
<td>00023500, 00088700, 00093200, 00094100</td>
</tr>
</tbody>
</table>

Note that increasing the size of a table in such a way that the number of digits required to index it is increased (e.g., a change from 99 to 101) may require that the PICTURE clause for data items used as indices (subscripts) to that table may have to be changed (e.g., from S99 to S999). Note also that in some cases a dummy (terminator) entry is actually stored in the table.

REPLACING THE ALPHABET TABLE SEARCH BY DIRECT INDEXING

The first step in translating the current contents of the buffer is to find the alphabet table entry for the leftmost character in the buffer. In order to preserve machine-independence, DOTSYS III does this by searching the alphabet table linearly from the top for a match with the leftmost character in the buffer. If the alphabet table has been ordered with the higher-frequency items nearer the top, an average of about nine entries is tested. On the IBM 360/50, that takes about one millisecond, or roughly fifteen per cent of the average time spent per character.

On some machines, including the IBM 360, it is possible to reduce drastically the time required to find the appropriate alphabet table entry, by using the test character itself as an index. This is done by moving the character to a data item defined as USAGE DISPLAY (the default case) but redefined as USAGE COMPUTATIONAL. For example, on the IBM 360, the data item below describes a half-word integer whose value is determined by the character moved into its right-hand byte; its left-hand byte is binary zeroes.
01 CHARACTER-INDEX.
02 LEFT PICTURE X, VALUE LOW-VALUE
02 RIGHT PICTURE X.
01 N REDEFINES CHARACTER-INDEX,
      PICTURE S999, USAGE COMPUTATIONAL.

Thus, if a character is moved to RIGHT, then N is a number determined by that character, and can be used as an index into an array of pointers to the appropriate places in the alphabet table. For example, suppose the array of pointers is called ARRAY-OF-POINTERS and the left most character in the buffer is a space, whose alphabet table entry comes first. Moving the space to RIGHT gives N a value of hexadecimal 40 or decimal 64; the 64th entry in ARRAY-OF-POINTERS should be 1 since the space entry is first in the alphabet table. Thus, the two sentences,

MOVE RL1 TO RIGHT.

MOVE ARRAY-OF-POINTERS (N) TO LETTER.

replace the alphabet table search, provided that ARRAY-OF-POINTERS has been properly initialized, which takes only two sentences in the READ-ALPHABET loop during initialization. The same thing can be done to replace the search occurring in the paragraphs following TEST-NON-TERMINAL (which check right context) in the translation section.

Note that with indexing, there is no longer any need to order the alphabet table according to frequency; however, the order of contraction table sections must still match the order of the associated alphabet table entries.

The price paid for the reduced search time is an increased storage requirement, since ARRAY-OF-POINTERS must have $2^k$ entries, where $k$ is the number of bits in a character. On the IBM 360, ARRAY-OF-POINTERS would take up $2^8 \times 2 = 512$ bytes.
CONTRACTION TABLE SEARCH ALGORITHM

General Rationale

A form of tree search has been implemented for comparison of a string against the contraction table. This algorithm is inherently much more efficient than a linear search for any size table. Moreover, the proportionate cost (in time) for new entries is reduced: the time increases only logarithmically, rather than directly.

The method takes advantage of the fact that often a comparison failure on, say, the third character implies that quite a few additional entries (with the identical first three characters) can be skipped around. The method also conserves space in that only one numeric field must be added to each table entry to support the algorithm.

The basic idea is quite simple. In a given initial-letter section of the table, all the entries which start a new subsection (wherein the first two letters are identical) are chained together, using the added "branch" field as a forward link. This forms the "level 1" chain; a level 2 chain may be formed and so forth. The implicit structure is that of a binary tree. During search, one either follows the branch (continued on the current chain) if comparison failure occurs at the character whose index equals the current level, or else goes to the next entry and one level higher.

The main complication with this process is that, having reached the highest point in the tree and still not having found a match, it still may be necessary to return to the lower level. For example, "AB" may be at level 2 and would probably follow "ABCDE"—at, say, level 4—in the table. The key "ABCDF" would thus reach at least level 4 and yet may not actually match until the "AB" is encountered. The handling of this situation in the algorithm is facilitated by indicating the level of the next entry whenever the current level chain ends. This level is entered in the branch field, in negative form so as not to be confused with a forward pointer and also to indicate that the forward pointer is null.

It should be noted that the table must be properly ordered on input in order for the algorithms to work. The proper ordering condition is that entries whose first p letters correspond must be together in the table; formally, using the notation defined below: if there exist two entries, with indices q and r (q < r) and an integer p > 0 such that Cqj = Crj for all j in the range 1 ≤ j ≤ p, then for all t in the range q ≤ t ≤ r it is also true that Cqj = Ctj for all j in the range 1 ≤ j ≤ p.
Notation

The notation used in the flow charts (Figures 5 and 6) is as follows:

LET

\( C_{ij} \) be the \( j^{th} \) character in the \( i^{th} \) entry of the contraction table.

\( b_i \) be the value of the branch field for the \( i^{th} \) entry.

\( s_k \) be the index of the first entry in the contraction table for the \( k^{th} \) initial letter.

\( e_k \) be the index of the last entry for the \( k^{th} \) initial letter.

\( L \) be the index for the lowest entry to be considered at the current level.

\( H \) be the index for the highest entry to be considered at the current level.

\( V \) be the highest index for the current initial letter.

\( n \) be the current level.

\( A_n \) be the upper index bound for the \( n^{th} \) level.

\( m \) be the number of entries in the contraction table, not counting the extra "end of table" entry.

\( M \) be the number of initial letters.

\( w_j \) be the \( j^{th} \) character in the current input string (string being looked up).
Start

For $i = 1$ \((1)\) $m + 1$: $b_i \leftarrow 0$

$k \leftarrow 1$

$n \leftarrow 1$

$L \leftarrow s_k$

$H \leftarrow e_k$

$V \leftarrow H$

$i \leftarrow L + 1$

Yes $-$ $i > H$?

$C_{Ln} = \#? \text{ Yes}$

Yes $-$ $i > H$?

$C_{Ln} = C_{Ln} \text{ No}$

no $-$ $i = L + 1$?

$b_L \leftarrow (-b_i)$

$L \leftarrow L + 1$

$b_i \leftarrow n$

$b_L \leftarrow i$

$b_i \leftarrow n$

$n \leftarrow n + 1$

$L \leftarrow L + 1$

$H \leftarrow i - 1$

$L > H ? \text{ No}$

$n \leftarrow n - 1$

$n = 0? \text{ Yes}$

\[
A_n = \begin{cases}
A_n-1, & n > 1 \\
V, & n \leq 1
\end{cases}
\]

\[
H = \begin{cases}
A_n-1, & n > 1 \\
V, & n \leq 1
\end{cases}
\]

$A_n \leftarrow i$

$n \leftarrow n + 1$

$b_L \leftarrow (-n)$

$L \leftarrow L + 1$

$k \leftarrow k + 1$

$k > M? \text{ No}$

Done

Figure 5. Set-Up Algorithm

41
Figure 6. Look-Up Algorithm
APPENDIX I

LISTING OF COMPLETE IBM 360 JOB DECK

//C08 EXEC PGM=IEQCAL00,REGION=86K,
// PARM='ONEASIS,NCCPY,SEQ,CLIST,NCMAP,DECK'
//SYSUT1 DD UNIT=SYSCA,SPACE=(46C,1700,1001)
//SYSUT2 DD UNIT=SYSCA,SPACE=(46C,700,1001)
//SYSUT3 DD UNIT=SYSCA,SPACE=(46C,700,1001)
//SYSUT4 DD UNIT=SYSCA,SPACE=(46C,1700,1001)
//SYSLIN DD DSNAME=ELLOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(80,500,1001),DCB=BLKSIZE=80
//SYSPRINT DD SYSOUT=A,DCB=RECFM=FBA,RECL=121,BLKSIZE=121)
//SYSPUNCH DD SYSOUT=A,DCB=BLKSIZE=80
//SYSSYN CO *
//SYSSYN ID 'COTSYS3'
//AUTHOR M. R. GERMART, J. K. MILLEN, J. E. SULLIVAN
//INSTALLATION MIT, MITRE
//DATE-WRITTEN AUGUST 1570
//DATE-COMPILED 1970
//REMARKS
//FIRST REVISION OCTOBER 1970.
//CHANGE IN STACKING LOGIC FOR UNITS-OF-MEASURE REVERSAL.
//PERMIT DYNAMIC SELF-CHECKING SWITCH.
//SEVERAL MINOR FIXES AND MODIFICATIONS.
//ENVIRONMENT DIVISION.
//CONFIGURATION SECTION.
//SOURCE-COMPUTER IBM-360
//OBJECT-COMPUTER IBM-360
//INPUT-OUTPUT SECTION.
//FILE-CONTROL.
// SELECT SYSINLUT, ASSIGN TO 'SYSIN' UTILITY.
// SELECT SYSPRINT, ASSIGN TO 'SYSPRINT' UTILITY.
// SELECT SYSPUNCH, ASSIGN TO 'SYSPUNCH' UTILITY.
// SELECT SYSBRL, ASSIGN TO 'SYSBRL' UTILITY.
// SELECT SYSRPQ, ASSIGN TO 'SYSRPQ' UTILITY.
//DATA DIVISION.
//FILE SECTION.
//FD SYSINLUT
// LABEL RECORDS ARE STANDARD.
//RECORDING MODE IS F.
//RECORD CONTAINS 80 CHARACTERS
//BLCK CONTAINS 0 RECORDS
//DATA RECORD IS INPUT-RECORD.
//01 INPUT-RECORD.
//C2 TEXT.
//C3 CHAR OCCURS 80 TIMES, PICTURE X.
//C2 TABLE-INFO REDEFINES TEXT.
//C3 FIELD1.
04 FIELD1  PICTURE X(11).  00004700
04 FIELD2  PICTURE X(9).  00004800
03 FIELD3WHOLE REDEFINES FIELD1.
04 FIELD3  PICTURE X(10).  00004900
03 FILLER  PICTURE X(5).  00005100
03 FIELD4  PICTURE X.  00005200
03 FILLER  PICTURE A.  00005300
03 FIELD5  PICTURE 99.  00005400
03 FILLER  PICTURE A.  00005500
03 FILLER  PICTURE A.  00005600
03 FILLER  PICTURE A.  00005700
03 FIELDS.  00005800
04 FIELD6  PICTURE 999.  00005900
04 FIELD6  PICTURE 999.  00006000
04 FIELD6  PICTURE 999.  00006100
04 FIELD6  PICTURE 999.  00006200
04 FIELD6  PICTURE 999.  00006300
03 FILLER  PICTURE X(37).  00006400
03 CARD-XX  PICTURE 9999999.  00006410
00006500
00006600
00006700
00006800
00006900
00007000
00007100
00007200
00007300
00007400
00007500
00007600
00007700
00007800
00007900
00008000
00008100
00008200
00008300
00008400
00008500
00008600
00008700
00008800
00008900
00009000
00009100
00009200
00009300
00009400
00009500
00009600
00009700
00009800
C7 STACK-INDICATOR PICTURE $99, USAGE COMPUTATIONAL. 0015100
C7 SAVE-CURRENT PICTURE $99, USAGE COMPUTATIONAL. 0015120
C7 TAB-CL PICTURE $99, USAGE COMPUTATIONAL. 0015150
C7 CARDS-PRINTED PICTURE $99, USAGE COMPUTATIONAL. 0015310
C7 POETRY-INDICATOR PICTURE $9, USAGE COMPUTATIONAL. 0015400
C7 SELF-TEST PICTURE $9, USAGE COMPUTATIONAL. 0015500
C7 CC-HIGH PICTURE $99, USAGE COMPUTATIONAL, VALUE 80. 0015600
C7 SC-HIGH PICTURE $99, USAGE COMPUTATIONAL, VALUE 90. 0015700
C7 U+50-FREE-STACK PICTURE $99, USAGE COMPUTATIONAL. 0015800
C7 CURRENT-TYPE PICTURE $9, USAGE COMPUTATIONAL. 0015900
C7 CURRENT-TRANS PICTURE $99, USAGE COMPUTATIONAL. 0016000
C7 NXT-LEVEL PICTURE $9, USAGE COMPUTATIONAL. 0016100
C7 HS PICTURE $99, USAGE COMPUTATIONAL. 0016200
C7 CS PICTURE $99, USAGE COMPUTATIONAL. 0016300
C7 TS PICTURE $99, USAGE COMPUTATIONAL. 0016400
C7 HS PICTURE $99, USAGE COMPUTATIONAL. 0016500
C7 X PICTURE $99, USAGE COMPUTATIONAL. 0016600
C7 HIGH-NO PICTURE $99, USAGE COMPUTATIONAL. 0016700
C7 LOW-NO PICTURE $99, USAGE COMPUTATIONAL. 0016800
C7 NCS PICTURE $99, USAGE COMPUTATIONAL. 0016900
C7 TRANS-CLASS PICTURE $99, USAGE COMPUTATIONAL. 0017000
C7 XYZ PICTURE $99, USAGE COMPUTATIONAL. 0017100
C7 SPACE-COUNT PICTURE $99, USAGE COMPUTATIONAL. 0017200
C7 CURRENT-TRANS PICTURE $99, USAGE COMPUTATIONAL. 0017300
C7 L-NXCHR PICTURE $999, USAGE COMPUTATIONAL. 0017400
C7 PRIOR-SPACE PICTURE $9, USAGE COMPUTATIONAL. 0017500
C7 IN-CONTRACT PICTURE $9, USAGE COMPUTATIONAL. 0017600
C7 C-LOW PICTURE $99, USAGE COMPUTATIONAL. 0017700
C7 C-HIGH PICTURE $99, USAGE COMPUTATIONAL. 0017800
C7 C-NO PICTURE $999, USAGE COMPUTATIONAL. 0017900
C7 BRL-E-I PICTURE $99, USAGE COMPUTATIONAL. 0018000
C7 N-O-E-T PICTURE $99, USAGE COMPUTATIONAL. 0018100
C7 XX PICTURE $999, USAGE COMPUTATIONAL. 0018200
C7 NCT PICTURE $999, USAGE COMPUTATIONAL. 0018300
C7 PAGINATION PICTURE X*. 0018400
C7 SEQ-ERR-IND PICTURE X. 0018400
C7 PREV-CARD-NO PICTURE 9999999. 0018420
C7 CARD-NO PICTURE 9999999. 0018440
C7 SC-ERROR-CT PICTURE 999. 0018450
C7 HEUD-INO PICTURE 9*.
O1 BUFFER.
O2 L.
O3 RL9 PICTURE X(9). 0018500
O3 RLI PICTURE X. 0018700
O3 R redefine L.
O3 RL1 PICTURE X. 0018900
O3 RRS PICTURE X(9). 0019000
C2 RCAR REDEFINES L.
C2 FIRSTCHAR PICTURE X. 0019100
C2 RCAR OCCURS 9 TIMES, PICTURE X. 0019300
C2 TAR-ECH REDEFINES L.
C2 FIRSTCHARS PICTURE 99. 0019500
C2 SCNOCHARS PICTURE 99. 0019600
C2 SCNOCHARS PICTURE 99. 0019700
01 RIGHT-CONTEXT-TABLE.
 02 RIGHT-CONTEXT-TABLE-ENTRY OCCURS 2 TIMES.
 03 NON-TERMINAL PICTURE X.
 03 RIGHT-CONTEXT-CLASSES.
 04 RIGHT-CONTEXT-CLASS OCCURS 4 TIMES.
 04 PICTURE 599, USAGE COMPUTATIONAL.

01 DECISION-TABLE.
 02 DECISION-TABLE-COLUMN OCCURS 15 TIMES.
 03 DECISION OCCURS 25 TIMES, PICTURE X.
 03 CONDITION OCCURS 10 TIMES, PICTURE X.

01 TRANSITION-TABLE.
 02 TRANSITION-TABLE-COLUMN OCCURS 10 TIMES.
 03 TRANSITION OCCURS 25 TIMES, PICTURE X.

01 SIGN-TABLE.
 02 SIGN-TABLE-ENTRY OCCURS 64 TIMES.
 03 OCTS-1-4 PICTURE XX.
 03 OCTS-2-5 PICTURE XX.
 03 OCTS-3-6 PICTURE XX.
 03 PROOF-CHARACTERS PICTURE XXX.

01 CLIP-TAB-WORK-AREA.
 02 BRAILLE-LINES.
 03 BRAILLE-LINE OCCURS 3 TIMES.
 04 BRAILLE-TEXT PICTURE X1201.
 02 BRAILLE-LINES-INOEXEO REFINES BRAILLE-LINES.
 03 BRAILLE-LINE-INOEXEO OCCURS 3 TIMES.
 04 BRAILLE-SIGN OCCURS 40 TIMES, PICTURE XXX.
 02 BRAILLE-LINE-CHARS REFINES BRAILLE-LINES.
 03 LINE-CHARS OCCURS 3 TIMES.
 04 LINE-CHAR OCCURS 120 TIMES, PICTURE X.
 02 PROOF-LINE.
 03 PROOF-TEXT PICTURE X1201.
 02 OUTPUT-TEXT REFINES PROOF-TEXT.
 04 OUTPUT-COLUMN OCCURS 120 TIMES, PICTURE X.
 03 OUTPUT-COLUMN-PAIRS REFINES PROOF-TEXT.
 04 COLUMN-PAIR OCCURS 60 TIMES, PICTURE 99.
 02 PROOF-LINE-INOEXEO REFINES PROOF-LINE.
 03 PROOF OCCURS 40 TIMES, PICTURE XXX.
 02 CCOE-LINE.
 03 CCOE-TEXT PICTURE X1801.
 02 CCOE-LINE-INOEXEO REFINES CCOE-LINE.
 03 CCOE-SIGN OCCURS 40 TIMES, PICTURE 99.
 02 TITLE-LINES.
 03 TITLE-LINE OCCURS 3 TIMES.
 04 TITLE-SIGN OCCURS 40 TIMES, PICTURE XXX.
 02 TITLE-PROOF.
 03 TITLE-PROOF-SIGN OCCURS 40 TIMES, PICTURE XXX.
 02 TITLE-SIGN-COE.
 03 TITLE-SIGN-COE OCCURS 40 TIMES, PICTURE 99.
LETTER-TO-DIGIT-CODE.
DIGIT-SIGNS.
  00 PICTURE S99, USAGE COMPUTATIONAL, VALUE 26.
  01 PICTURE S99, USAGE COMPUTATIONAL, VALUE 1.
  02 PICTURE S99, USAGE COMPUTATIONAL, VALUE 3.
  03 PICTURE S99, USAGE COMPUTATIONAL, VALUE 9.
  04 PICTURE S99, USAGE COMPUTATIONAL, VALUE 25.
  05 PICTURE S99, USAGE COMPUTATIONAL, VALUE 17.
  06 PICTURE S99, USAGE COMPUTATIONAL, VALUE 11.
  07 PICTURE S99, USAGE COMPUTATIONAL, VALUE 27.
  08 PICTURE S99, USAGE COMPUTATIONAL, VALUE 19.
  09 PICTURE S99, USAGE COMPUTATIONAL, VALUE 10.
DIGIT-SIGNS-INDEXED REDEFINES DIGIT-SIGNS.
  03 DIGIT OCCURS 10 TIMES, PICTURE S99,
    USAGE COMPUTATIONAL.

PAGE-NO-HEAD.
CRT-PAGE PICTURE S999.
CRT-DIGIT REDEFINES CRT-PAGE.
  CRT-PAGE-DIGIT PICTURE 9, OCCURS 4 TIMES.

CONTRACT-CHILD-STRUCT.
  C-F-CHILD PICTURE S99, USAGE COMPUTATIONAL.
  CONTRACT-CHILD-WHOLE.
   CONTRACT-CHILD PICTURE X(110).
  C-F-LETTERS REDEFINES CONTRACT-CHILD-WHOLE.
   CONTRACT-CHILD-CHAR OCCURS 10 TIMES, PICTURE X.

CONTRACTION-LIST.
  LIST-ENTRY OCCURS 189 TIMES.
  INKPRINT PICTURE X(110).
  BRaille-EQUIV OCCURS 4 TIMES, PICTURE S99,
    USAGE COMPUTATIONAL.

PROCEDURE DIVISION.

INITIALIZATION SECTION.

OPEN-FILES.
  OPEN INPUT SYSIN.
  OPEN DLPTL SYSPRINT.
  OPEN OUTPUT SYSOUT.
  OPEN OUTPUT SYSERR.
  OPEN OUTPUT SYSPRINT.
  MOVE SPACES TO OUT.
  WRITE OUTPUT-LINE AFTER ADVANCING 0.
  MOVE SPACES TO OUT-BRL.
  WRITE OUTPUT-BRL AFTER 0.
  MOVE SPACES TO OUT-RPQ.
  MOVE OUTPUT-RPQ AFTER 0.
  MOVE 'N' TO SEQ-ERR-IND.
  MOVE ZERO TO PREV-CARD-NO.
MOVE 'E' TO ECHO.*
PERFORM READ-CARO.*
MOVE FIELC10 TO ECHO.*
MOVE 0 TC N.*

LOOP1.*
ADD 1 TC N.
MOVE 0 TO NO-OF-ELEMENTS (N).*
MOVE 0 TO SPECIAL-CONTROL (N).*
COMPUTE PREVIOUS-STACK (N) = N - 1.*
COMPUTE NEXT-STACK (N) = N + 1.*
IF N NOT = 15 THEN GO TO LOOP1.*
MOVE 0 TO NEXT-STACK (19).*
MOVE 0 TO NEXT-STACK (1).*
MOVE 0 TO PREVIOUS-STACK (2).*
MOVE 3 TO HEAD-FREE-STACK.*
MOVE 1 TO CURRENT-TYPE.*
MOVE 1 TO HEAD-S (1).*
MOVE 1 TO TAIL-S (1).*
MOVE 1 TO CURRENT-S (1).*
MOVE 2 TO TAIL-S (2).*
MOVE 2 TO CURRENT-S (2).*
MOVE 2 TO HEAD-S (2).*
MOVE 0 TO PREVIOUS-STACK (3).*
MOVE 0 TO NEXT-STACK (2).*
MOVE 0 TO N.*

LOOP2.*
ADD 1 TC N.
MOVE 'L' TO TAB-TYPE (N).*
COMPUTE TAB-COLUMN (N) = 5 * N.*
IF N NOT = 9 THEN GO TO LOOP2.*
MOVE 0 TO LINECOUNT.*
MOVE 0 TO NALPHABET.*
MOVE 1 TO OUTPTR.*
MOVE 0 TO TABULATE.*
MOVE 0 TO SPACE-COUNT.*
MOVE 0 TO XX.*
MOVE 0 TO HEAD-INO.*
MOVE 'X' TO TEMPL.*
MOVE 2 TO STACK-INDICATOR.*
MOVE 0 TO SC-ERROR-CT.*
MOVE 0 TO SELF-TEST.*
MOVE 0 TO CARCS-PRINTED.*
MOVE 1 TO CURRENT-IN-TRANS.*
MOVE 0 TO PRIOR-SPACE.*
MOVE 0 TO IN-CONTRACT.*
ALTER S-T-B1 TO PROCEED TO S-T-NC1.*
MOVE 0 TO N.*
MOVE 0 TO N.*

LOOP1.*
ADD 1 TC N.
MOVE 0 TO NO-OF-ELEMENTS-TRANS (N).*
IF N < 4 GO TO LOOP2.*
MOVE 0 TO CURRENT-TRANS.*
MOVE 0 TO N.*
```plaintext
LOOP2A.
  PERFORM READ-CARO.
  ADD 1 TC N.
  MOVE FIEL01 TO IAKPRINT (N).
  MCVE FIEL051 TO BRAILLE-EQUIV (N, 1).
  MOVE FIEL052 TO BRAILLE-EQUIV (N, 2).
  MOVE FIEL053 TO BRAILLE-EQUIV (N, 3).
  MCVE FIEL054 TO BRAILLE-EQUIV (N, 4).
  IF N < 189 THEN GO TO TOP LOOP2A.
  MOVE 999 TO EXTENT (1).
  MOVE 1 TC N.
READ-ALPHABET.
  MOVE EXTENT (1) TO EXTENT (N).
  PERFORM READ-CARO.
  ADD 1 TO NALPHABET.
  IF FIEL03 = *99' THEN GO TO LOOP63.
  MOVE FIEL011 TO SYMBOL (N).
  MOVE FIEL03 TO CHAR-CLASS (N).
  MOVE FIEL051 TO SINGLE-SIGN (N).
  MOVE FIEL054 TO ISC-LETTER (N).
  MOVE FIEL04 TO COMP-B (N).
  ADD 1 TO N.
  GO TO READ-ALPHABET.
LOOP63.
  MCVE 1 TO I.
  MOVE 1 TO N.
FILL-TABLE.
  PERFORM READ-CARO.
  IF FIEL011 = TEMPL THEN GO TO MOVE-FIELDS.
  MOVE FIEL01 TO TEMPL.
  MOVE N TO EXTENT (1).
  IF SYM8CL (1) = TEMPL THEN GO TO ACOII.
  IF FIEL03 = *99' THEN GO TO ACOII.
  MCVE SPACES TO OUT.
  MOVE 'E' TO ECHO.
  MOVE *** TABLE SEQUENCE ERROR *** TO OUT.
  WRITE OUTPUT-LINE AFTER ADVANCING 1.
  MOVE 'Y' TO SEQ-ERR-INO.
ACOII.
  ADD 1 TC I.
MOVE-FIELDS.
  MCVE FIELC12 TO STRING (N).
  MOVE FIEL02 TO RIGHT-CONTEXT (N).
  MOVE FIEL03 TO INPUT-CLASS (N).
  MOVE FIEL04 TO SHIFT (N).
  MOVE FIEL051 TO SIGN (N, 1).
  MOVE FIEL052 TO SIGN (N, 2).
  MOVE FIEL053 TO SIGN (N, 3).
  MOVE FIEL054 TO SIGN (N, 4).
  ADD 1 TC N.
  IF FIEL03 NOT = *99' THEN GO TO FILL-TABLE.
```

COMPUTE NCT = N - 2.
COMPUTE J = I - 1.
COMPUTE EXTENT (I) = EXTENT (J) + 1.
MOVE J TO MAXEXTENT.
MCVE 1 TO I.
LABEL0
MOVE ZERO'S TO BRANCH (I).
ADD 1 TO I.
IF I < NCT + 2 THEN GC TO LABEL1.
MOVE 1 TO I.
LABEL0
MOVE I TO N.
MOVE EXTENT (I) TO LOW-NO.
COMPUTE J = I + 1.
COMPUTE HIGH-NO = EXTENT (J) - I.
MOVE HIGH-NO TO VHIGH.
LABEL0.
COMPUTE II = LOW-NO + 1.
IF II > HIGH-NO THEN GO TO LABEL5.
IF TABLE-CHAR (LOW-NC, N) = RCC THEN GO TO LABEL2.
LABEL0.
IF II > HIGH-NO THEN GO TO LABEL5.
IF TABLE-CHAR (LOW-NO, N) NOT = TABLE-CHAR (II, N)
THEN GO TO LABEL2.
ADD 1 TO II.
GO TO LABEL11.
LABEL2.
MOVE II TO L-ARRAY (N).
MOVE II TO BRANCH (LOW-NO).
MCVE N TO BRANCH (II).
ADD 1 TO N.
ADD 1 TO LOW-NO.
COMPUTE HIGH-NO = II - 1.
LABEL3.
IF LOW-NO NOT > HIGH-NO THEN GO TO LABEL1.
SUBTRACT 1 FROM N.
LABEL4.
IF N = 0 THEN GO TO LABEL6.
COMPUTE K = N - 1.
IF N > 1 THEN COMPUTE HIGH-NO = L-ARRAY (K) - 1
ELSE MOVE WHIGH TO HIGH-NO.
GO TO LABEL3.
LABEL5.
IF II NOT = LOW-NO + 1 THEN GO TO LABEL9.
COMPUTE BRANCH (LOW-NC) = -BRANCH (II).
ADD 1 TO LOW-NO.
MCVE N TO BRANCH (II).
GO TO LABEL4.
LABEL5.
MOVE II TO L-ARRAY (N).
ADD 1 TO N.
COMPUTE BRANCH (LCW-NC) = - N.
ADD 1 TO LOW-NO.
ADD 1 TO LOW-NO.
GC TO LABEL1.
LABEL1.
AD1 TC I.
IF I > MAXEXTENT THEN GO TO READ-RIGHT-CONTEXT-TABLE.
GC TO LABEL10.
READ-RIGHT-CONTEXT-TABLE.
PERFORM READ-CARO.
MOVE FIELO3 TO NNT.
MOVE 1 TO N.
READ-CONDITIONS.
ADD 1 TC J.
MOVE CHAR II TO CONDITION (N; J).
IF I < NIC + NSV THEN GO TO READ-CONDITIONS.
ADD 1 TC A.
IF N NOT > NNT THEN GO TO READ-DECISION-TABLE.
MOVE PRIMARY-STATE TO STATE-VECTOR.
READ-SIGN.
PCVE 1 TO N.
READ-SIGN-ENTRY.
PCVE 1 TO N.
READ-SIGN-ENTRY.
PERFORM READ-CARO.
MOVE TABLE-INFO TO SIGN-ENTRY (N).
ADD 1 TC N.
IF N ACT > 64 THEN GO TO READ-SIGN-TABLE-ENTRY

READ-COUNT-CARCS.
PERFORM READ-CARCS.
MOVE FIELD10 TO PROOF-OUT.
PERFORM READ-CARCS.
MOVE FIELD10 TO BRAIMLE.
PERFORM READ-CARCS.
MCVE FIELD10 TO RFC.
PERFORM READ-CARCS.
MOVE FIELD10 TO PUNCH-O-OUTPUT.
PERFORM READ-CARCS.
MCVE FIELD3 TO OUTL.
PERFORM READ-CARCS.
MOVE FIELD3 TO LPG.
IF PUNCH-O-OUTPUT NCT = 'N' THEN OPEN OUTPUT PUNCH.
PERFORM READ-CARCS.
MOVE FIELD6 TO CRT-PAGE.
MOVE FIELD11 TO PAGINATION.

TABLE-DISPLAY.
IF PROOF-O-OUT NOT = 'P' THEN GO TO SKIP-DISPLAY.
MCVE SPACES TO OUT.
WRITE OUTPUT-LINE AFTER 0.
MOVE 'RIGHT CONTEXT TABLE' TO OUT.
WRITE OUTPUT-LINE AFTER 2.
MOVE 'NON-TERMINAL INPUT CLASSES' TO OUT.
WRITE OUTPUT-LINE AFTER 2.
MOVE 1 TO K.

DISP14. MOVE SPACES TO OUTPUT-TEXT.
MCVE NON-TERMINAL (K) TO OUTPUT-COLUMN (6).
MCVE RIGHT-CONTEXT-CLASS (K, 1) TO COLUMN-PAIR (8).
MOVE RIGHT-CONTEXT-CLASS (K, 2) TO COLUMN-PAIR (10).
MCVE RIGHT-CONTEXT-CLASS (K, 3) TO COLUMN-PAIR (12).
MCVE RIGHT-CONTEXT-CLASS (K, 4) TO COLUMN-PAIR (14).
MCVE OUTPUT-TEXT TO OUT.
WRITE OUTPUT-LINE AFTER 2.
MOVE 1 TO K.
IF K NOT > NCT THEN GO TO DISP14.

DISP13. MOVE SPACES TO OUT.
WRITE OUTPUT-LINE AFTER 0.
MOVE 'DECISION TABLE' TO OUT.
WRITE OUTPUT-LINE AFTER 3.
MOVE 'COLUMN' TO OUTPUT-TEXT.
PERFORM DISP1 VARYING K FROM 1 BY 1 UNTIL K = NOTC.

DISP1. COMPUTE I = 9 + 2 * K.
MCVE K TO COLUMN-PAIR (I).
ENO-DISP1.
MCVE OUTPUT-TEXT TO OUT.
WRITE OUTPUT-LINE AFTER 2.
MOVE SPACES TO OUT.
WRITE OUTPUT-LINE AFTER 1.

DISP5.
MOVE I TC N.
MOVE 'INPUT CLASS' TO OUTPUT-TEXT.

DISP6.
MOVE \( \Lambda \) TO COLUMN-PAIR (9).
MOVE I TO K.

DISP7. COMPUTE \( I = 18 + 4 \times K \).
MOVE DECISION (K, \( \Lambda \)) TO OUTPUT-COLUMN (1).
ADD 1 TO K.
IF K NOT > NIC THEN GO TO DISP8.
MOVE OUTPUT-TEXT TO OLT.
MOVE SPACES TO OUTPUT-TEXT.

DISP8. WRITE OUTPUT-LINE AFTER 1.
ADD 1 TO N.
IF N NOT > NIC THEN GC TO DISP6.
MOVE SPACES TO OUT.
WRITE OUTPUT-LINE AFTER 1.
MOVE I TC K.
MOVE 'STATE VARIABLE' TO OUTPUT-TEXT.

DISP9. MOVE \( \Lambda \) TO COLUMN-PAIR (9).
MOVE I TO K.

DISP10. COMPUTE \( I = 18 + 4 \times K \).
MOVE CONDITION (K, N) TO OUTPUT-COLUMN (1).
ADD 1 TO K.
IF K NOT > NIC THEN GC TO DISP10.
MOVE OUTPUT-TEXT TO OUT.
MOVE SPACES TO OUTPUT-TEXT.

DISP11. WRITE OUTPUT-LINE AFTER 1.
ADD 1 TO N.
IF N NOT > NSV THEN GC TO DISP2.

DISP12. MOVE SPACES TO CUT.
WRITE OUTPUT-LINE AFTER 0.
MOVE 'TRANSITION TABLE' TO OUT.
WRITE OUTPUT-LINE AFTER 3.
MOVE 'STATE VARIABLE' TO OUTPUT-TEXT.
PERFORM DISP1 VARYING K FROM 1 BY 1 UNTIL K > NSV.
MOVE OUTPUT-TEXT TO OLT.
WRITE OUTPUT-LINE AFTER 2.
MOVE SPACES TO OUT.
WRITE OUTPUT-LINE AFTER 4.
MOVE 'INPUT CLASS' TO OUTPUT-TEXT.
PERFORM DISPI10 THRU DISPI11 VARYING N FROM 1 BY 1.
LATIL N = NIC.

DISP13. MOVE \( \Lambda \) TO COLUMN-PAIR (9).
MOVE I TO K.

DISP14. COMPUTE \( I = 18 + 4 \times K \).
MOVE TRANSITION (K, N) TO OUTPUT-COLUMN (1).
ADD 1 TO K.
IF K NOT > NSV THEN GO TO DISPI2.
MOVE OUTPUT-TEXT TO CUT.
MCVE SPACES TO OUTPUT-TEXT.
DISPIL. WRITE OUTPUT-LINE AFTER 1.
END-DISPIL. MOVE SPACES TO OUT.
WRITE OUTPUT-LINE AFTER 0.
SKIP-DISPIL.

IF SEC-ERR-IND = 'Y' THEN GO TO CLOSE-FILES.
READ-FIRST-RECORD.
  PERFORM READ-NEW-RECORD.
  PERFORM SHIFT-LEFT-CHE 10 TIMES.
  MCVE SPACES TO BRAILLE-LINES.
  MCVE SPACES TO PROOF-LINE.
  MOVE ZEROS TO CODE-LINE.
  GC TO TRANSLATION.
NTTHING. EXIT.
END-INITIALIZATION.

TRANSLATION SECTION.

INSPECT-BUFFER.
  MCVE L TO LETTER.
IDENTIFY-FIRST-CHARACTER.
  IF SYMBOL (LETTER) = RLL THEN GO TO LOOKUP.
  IF LETTER = NALPHA B LETTER THEN GO TO TEST-RCC.
  ADD L TO LETTER.
  GO TO IDENTIFY-FIRST-CHARACTER.

TEST-RCC.
  IF RLL NOT = RCC AND RLL NOT = '*' THEN GO TO ERROR-1.
  MOVE 50 TO OUTSIGN.
  PERFORM OUTPUT-SIGN.
  NOTE RUN WILL STOP WITH ABOVE PERFORM.

LOOKUP.
  IF ISO-LETTER (LETTER) NOT = 1 THEN GO TO LOOP80.
  IF (RLL = '1') AND ((RCHAR (0) = 'A') OR (RCHAR (0) = 'I')
          OR (RCHAR (1) = 'Q') CR (RCHAR (1) = 'E'))
      THEN GO TO LOOP80.
  MOVE 1 TO N.
  MOVE I TO K.

LOOP81.
  IF RCHAR (N) = '1' THEN GO TO LOOP82.
  IF RCHAR (K) = '1' THEN GO TO LOOP80.
  IF RCHAR (N) = '1' THEN GO TO LOOP88.
  IF RCHAR (N) = '1' THEN GO TO LOOP82.
  MOVE 1 TO X.

ICEN-CHAR.
  IF SYMBOL (X) = RCHAR (N) THEN GC TO LOOP83.
  ADD 1 TO X.
  GO TO ICEN-CHAR.

LOOP82.
IF CHAR-CLASS (X) NOT = 1 THEN GO TO LOOP80.
COMPUTE M = N + 1.
IF RLI = '1' THEN GC TO LOOP90.
IF SYMBOL (LETTER) = '(' THEN GO TO LOOP84.
IF SYMBOL (LETTER) = RCHAR (M) THEN GO TO LOOP85.
GC TO LOOP80.
LOOP84.
IF RCHAR (M) NOT = ')' THEN GC TO LOOP80.
LOOP85.
PERFORM SHIFT-CHAR K TIMES.
PERFORM SHIFT-LEFT-CNE M TIMES.
MCVE '=' TO RLI.
GC TO INSPECT-BUFFER.
LOOP88.
ADD 1 TO K.
LOOP82.
ADD 1 TO N.
GO TO LOOP81.
LOOP8C.
MCVE 1 TO X.
LOOP51.
IF SYMBOL (X) = RCHAR (M) THEN GC TO LOOP92.
ADD 1 TO X.
GC TO LOOP91.
LOOP52.
IF CHAR-CLASS (X) = 1 THEN GO TO LOOP80.
IF N > K THEN PERFORM SHIFT-LEFT-CNE.
MCVE '=' TO RLI.
GC TO INSPECT-BUFFER.
SHIFT-CHAR.
MCVE RCHAR (N) TO RCHAR (M).
SLETRACT I FROM N.
SLETRACT I FROM M.
END-SHIFT-CHAR.
LOOP8C.
COMPUTE NN = LETTER + 1.
MOVE EXTENT (LETTER) TO INDX.
IF INDX > NCT THEN GC TO MOVE-SINGLE-SIGN.
MCVE 1 TO NXT-LEVEL.
MATCH.
MOVE NXT-LEVEL TO POINTER.
COMPARE.
IF TABLE-CHAR (INDEX, POINTER) = RRC
THEN GO TO TEST-NON-TERMIN.
IF RCHAR (POINTER) NOT = TABLE-CHAR (INDEX, POINTER)
THEN GO TO COMPARE-FAIL.
ADD 1 TO POINTER.
IF POINTER > 9 THEN GC TO TEST-NON-TERMIN.
GC TO COMPARE.
MISMATCH.
COMPARE-FAIL.
IF BRANCH (INDX) > 0 THEN GC TO SKIP.
IF POINTER NOT = NXT-LEVEL THEN GO TO NO-MATCH-LEVEL.
LAMP1.
   IF - (BRANCH (INDEX)) < NXT-LEVEL THEN GO TO NO-MATCH-LEVEL.
       A00 1 TO INDEX.
LAMP2.
   IF BRANCH (INDEX) NOT > 0 THEN GC TO LAMP1.
       MOVE BRANCH (INDEX) TO INDEX.
       GC TO LAMP2.
NO-MATCH-LEVEL.
   COMPUTE NXT-LEVEL = - BRANCH (INDEX).
   IF NXT-LEVEL NOT > 1 THEN GO TO MOVE-SINGLE-SIGN.
       ADD 1 TO INDEX.
       GC TO MATCH.
SKIP.
   IF POINTER = NXT-LEVEL THEN GO TO JUMP.
   IF BRANCH (INDEX) NOT = INDEX + 1 THEN A00 1 TO NXT-LEVEL.
       A00 1 TO INDEX.
       GC TO MATCH.
JUMP.
   MOVE BRANCH (INDEX) TO INDEX.
       GC TO MATCH.
TEST-NON-TERMINAL.
   IF RIGHT-CONTEXT (INDEX) = SPACE THEN GO TO OUTPUT-LOGIC.
       MCVE 1 TO N.
TNT1.
   IF SYMBOL (IN) = RCAR (PCINTER) THEN GO TO TNT2.
       IF N = NALPHABET THEN GC TO MISMATCH.
       A00 1 TO N.
       GC TO TNT1.
TNT2.
   MOVE 1 TO K.
       IF RIGHT-CONTEXT (INDEX) NOT = NCA-TERMINAL (K)
           THEN GO TO TNT3.
TNT3.
   MOVE 1 TO J.
TNT4.
   IF CHAR-CLASS (IN) = RIGHT-CONTEXT-CLASS (K, J)
       THEN GO TO OUTPUT-LOGIC.
       IF RIGHT-CONTEXT-CLASS (K, J) = 99 THEN GO TO MISMATCH.
       ADD 1 TO J.
       IF J NOT > 4 THEN GO TO TNT4 ELSE GO TO MISMATCH.
TNT5.
   A00 1 TO K.
       IF K NOT > NNT THEN GO TO TNT5 ELSE GO TO MISMATCH.
OUTPUT-LOGIC.
   MOVE INPUT-CLASS (INDEX) TO I.
       MCVE I TO TRANS-CLASS.
       MCVE 1 TO K.
LOGIC.
   IF DECISION (K, I) = '-' THEN GO TO LOGIC4.
   IF DECISION (K, I) = 'G' THEN GO TO TRANS-340.
IF DECISION (K, I) = 'F' THEN GO TO MISMATCH.
MOVE 1 TO N.

LOGIC2.
IF CONDITION (K, N) = 'T' THEN GO TO LOGIC3.
IF STATE-VARIABLE (N) NCT = CONDITION (K, N)
THEN GO TO LOGIC5.

LOGIC3.
IF N = NSV THEN GO TO LOGIC5.
ADD 1 TO N.
GO TO LOGIC2.

LOGIC4.
IF K = NCT THEN GC TO MISMATCH.
ADD 1 TO K.
GO TO LOGIC1.

LOGIC5.
IF DECISION (K, I) = 'Y' THEN GO TO TRANSOUC,
ELSE GO TO MISMATCH.

LOGIC6.
IF DECISION (K, I) = 'Y' THEN GO TO TRANSOUC,
ELSE GO TO MISMATCH.

TRANSOUC.
MOVE SHIFT INDEX TO J.
PERFORM SHIFT-LEFT-CNE J TIMES.
MOVE 1 TO J.

PUT-OUT-SIGNS.
IF SIGN (INDEX, J) = 99 THEN GO TO END-OUTPUT-SIGNS.
MOVE SIGN (INDEX, J) TO CUTSIGN.
PERFORM DLTPUT-SIGN.
ADD 1 TO J.
IF J NOT > 4 THEN GO TO PUT-OUT-SIGNS.

END-OUTPUT-SIGNS.

TRANSITICA-LOGIC.
MOVE 1 TO N.
MOVE TRANS-CLASS TO I.

LOGIC7.
IF TRANSITION (N, I) = 'T' THEN NEXT SENTENCE,
ELSE IF TRANSITION (N, I) = 'S'
CR (TRANSITION (N, I) = 'T' AND STATE-VARIABLE (N) = 'N')
THEN MOVE 'Y' TO STATE-VARIABLE (N),
ELSE MOVE 'N' TO STATE-VARIABLE (N).
ADD 1 TO N.
IF N ACT > NSV THEN GC TO LOGIC6.
GO TO TRANSLATION.

MCVE-SINGLE-SIGN.
MOVE SINGLE-SIGN LETTER) TO OLTSIGN.
PERFORM OUTPUT-SIGN.
MOVE CHAR-CLASS (LETTER) TO TRANS-CLASS.
PERFORM SHIFT-LEFT-CNE.
GC TO END-OUTPUT-SIGNS.

ERROR-1.
MOVE 'NEW CHAR' TO CLI.
MOVE RL1 TO OUT-PLACE (12).
WRITE CLIPUT-LINE AFTER 2.
MOVE SPACES TO CL.
MOVE 'E-' TO RL1.
GO TO TRANSLATION.

INPUT-CHAR SECTION.

MCVE-NEXT-CHAR.
IF INPTR > 72 THEN PERFORM READ-NEW-RECORO.
MOVE CHAR INPTR) TO NXTCHR.
ADD 1 TO INPTR.
IF NXTCHR NOT = ' ' THEN GC TO RESET-PRIOR-SPACE.
IF PRIOR-SPACE = 0 THEN GO TO MOVE-NEXT-CHAR.
SUBTRACT 1 FROM PRIOR-SPACE.
GO TO S-T-81.
RESET-PRIOR-SPACE.
IF NXTCHR = ' ' THEN MOVE 2 TO PRIOR-SPACE, ELSE
MCVE 1 TO PRIOR-SPACE.
S-T-81.
GO TO.
MCVE SELF-TEST OPTION SWITCH.
S-T-80.
IF NXTCHR = bsc THEN GO TO MOVE-NEXT-CHAR,
ELSE GO TO END-INPUT.
S-T-YES.
IF IN-CONTACT = 1 THEN GO TO YES-IN-CONTACT.
NOT-IN-CONTACT.
IF NXTCHR = ' ' THEN GO TO SET-IN-CONTACT.
IF NXTCHR = ' ' THEN GO TO INC-C-I-T.
MOVE NO-OF-ELEMENTS-TRANS (CURRENT-IN-TRANS) TO N-O-E-T.
IF NO-O-E-T < 30 THEN AQC 1 TO N-O-E-T
MOVE A-O-E-T TO NC-OF-ELEMENTS-TRANS (CURRENT-IN-TRANS).
MCVE 1 TO L-NXTCHR.
LKP-NTCHR.
IF SYMBOL (L-NXTCHR) = NXTCHR THEN GC TO STORE1-E-T.
IF L-NXTCHR = NALPHABET THEN GO TO STORE1-E-T-99.
ADD 1 TO L-NXTCHR.
GO TO LKP-NTCHR.
STORE1-E-T-99.
MOVE 59 TO ELEMENT-TRANS (CURRENT-IN-TRANS, N-O-E-T)
GO TO END-INPUT.
STORE1-E-T.
MOVE SINGLE-SIGN (L-NXTCHR) TO ELEMENT-TRANS
(CURRENT-IN-TRANS, N-C-E-T).
GO TO END-INPUT.
SET-IN-CONTACT.
MCVE 0 TO C-F-COUNT.

MCVE SPACES TO CONTRACT-FORM.

MOVE 1 TO IN-CONTRACT.

GC TO MOVE-NEXT-CHAR.

YES-IN-CONTRACT.

IF NXTCHR = RCC THEN GO TO LKUP-CC.

IF NXTCHR = ' ' THEN GO TO LKUP-CC.

IF C-F-COUNT < 10 THEN ADD 1 TO C-F-COUNT.

MOVE NXTCHR TO CONTRACT-FORM-CHAR (C-F-COUNT).

GO TO END-INPUT.

LKUP-CC.

IF C-F-COUNT = 0 THEN GO TO RESET-IN-CONTRACT.

MOVE NO-OF-ELEMENTS-TRANS (CURRENT-IN-TRANS) TO N-O-E-T.

MOVE 1 TO C-L-LOW.

MOVE 185 TO C-L-HIGH.

C-L-BIN-SEARCH-LOOP.

COMPUTE C-L-CURR = (C-L-LOW + C-L-HIGH) / 2.

IF INPRINT (C-L-CURR) > CONTRACT-FORM THEN GO TO BOTT-HALF.

IF INPRINT (C-L-CURR) < CONTRACT-FORM THEN GO TO TOP-HALF.

GO TO C-L-FOUND.

BCTT-HALF.

IF C-L-LOW NOT < C-L-CURR THEN GO TO C-L-NOT-FOUND.

CCOMPUTE C-L-HIGH = C-L-CURR - 1.

GO TO C-L-BIN-SEARCH-LOOP.

TOP-HALF.

IF C-L-HIGH NOT > C-L-CURR THEN GO TO C-L-NOT-FOUND.

CCOMPUTE C-L-LOW = C-L-CURR + 1.

GO TO C-L-BIN-SEARCH-LOOP.

C-L-NOT-FOUND.

IF N-O-E-T < 30 THEN ADD 1 TO N-C-E-T.

MOVE N-O-E-T TO NO-OF-ELEMENTS-TRANS (CURRENT-IN-TRANS).

MOVE 99 TC ELEMENT-TRANS (CURRENT-IN-TRANS, N-O-E-T).

GO TO RESET-IN-CONTRACT.

C-L-FOUND.

MOVE 1 TO BRL-E-I.

MCVE 1 TO BRL-E-MOVE.

IF BRAILLE-EQUIV (C-L-CURR, BRL-E-I) = 99 THEN GO TO STORE-NOET.

IF N-O-E-I < 30 THEN ADD 1 TO N-O-E-T.

MOVE BRAILLE-EQUIV (C-L-CURR, BRL-E-I) TO ELEMENT-TRANS (CURRENT-IN-TRANS, N-O-E-T).

ACO 1 TO BRL-E-I.

IF BRL-E-I NOT > 4 THEN GO TO BRL-E-MOVE.

STORE-NOET.

MOVE N-O-E-T TO NO-OF-ELEMENTS-TRANS (CURRENT-IN-TRANS).

RESET-IN-CONTRACT.

MOVE 0 TO IN-CONTRACT.

IF NXTCHR = RCC THEN GO TO MOVE-NEXT-CHAR.

INC-C-I-T.

ADD 1 TO CURRENT-IN-TRANS.

IF CURRENT-IN-TRANS > 6 THEN MOVE 1 TO CURRENT-IN-TRANS.

MOVE 0 TO NO-OF-ELEMENTS-TRANS (CURRENT-IN-TRANS).

GO TO END-INPUT.
FIN.
    ACC 1 TC SPACE-COUNT.
    IF (SPACE-COUNT = 1) ANC (PRIOR-SPACE > 0)
        THEN MOVE ' ' TC NXTCHR.
        ELSE MOVE RGC TC NXTCHR.
    GO TO END-INPUT.
READ-NEW-RECORD.
    IF SPACE-COUNT > 0 THEN GO TO FIN.
    READ SYSINPUT, AT END GO TC FIN.
    MOVE I TO INPTR.
    IF PROOF-OUT NOT = 'P' THEN GC TC END-INPUT.
    MOVE SPACES TO OUT.
    MOVE TEXT TO CUT.
    WRITE OUTPUT-LINE AFTER ADVANCING 1.
    ADD 1 TO CARDS-PRINTED.
END-INPUT. EXIT.
OUTPLT-SIGN SECTION.
    MOVE CURRENT-S (CURRENT-TYPE) TO CS.
    MOVE HEAD-S (CURRENT-TYPE) TO HS.
    MOVE TAIL-S (CURRENT-TYPE) TO TS.
    IF OUTSIGN < 64 THEN GC TO ORD-CHAR-CUT.
    IF OUTSIGN = 64 THEN GC TO SPACE-CUT.
    IF OUTSIGN NOT > CC-HIGH THEN GC TO CARRIAGE-CONTROL.
    IF OUTSIGN NOT > SC-HIGH THEN GC TO STACK-CONTROL.
    GO TO MOD-MODE-CONTROL.
SPACE-OUT.
    IF SELF-TEST = 0 THEN GC TO SKIP-SELF-TEST.
    ADD 1 TO CURRENT-TRANS.
    IF CURRENT-TRANS > 6 THEN MOVE 1 TO CURRENT-TRANS.
    PERFORM TRANS-TEST.
    IF WORD-ERROR = 0 THEN GC TO SKIP-SELF-TEST.
    MOVE CURRENT-TRANS TO M.
LCOP4.
    ADD 1 TO CURRENT-TRANS.
    IF CURRENT-TRANS > 6 THEN MOVE 1 TO CURRENT-TRANS.
    IF M = CURRENT-TRANS THEN GC TO MARK-ERROR.
    PERFORM TRANS-TEST.
    IF WORD-ERROR = 0 THEN GC TO MARK-ERROR.
    GO TO LOOP4.
MARK-ERROR.
    ADD 1 TO SC-ERROR-CT.
    MOVE 'ISK' TO PROCF (40).
    PERFORM MCVE-ERROR 2 TIMES.
SKIP-SELF-TEST.
    IF STACK-INDICATOR = 2 THEN PERFORM CLEAR-TB,
        ELSE PERFORM STACK-TB;
    NOTE STACK-INDICATOR IS ALWAYS 2, 4, OR 5.
GO TO ENO-OLPUT. 00100700
OPO-CHAR-CUT. 00100800
IF NO-OF-ELEMENTS (CS) = 30
  THEN GC TO OVER-FLW-STACK. 00101000
  ADD 1 TC NO-OF-ELEMENTS (CS). 00101100
CONTINUE-OPO. 00101200
  MCVE NO-OF-ELEMENTS (CSI) TO NCS. 00101300
  MCVE OUTSIGN TO ELEMENT ICS, NCSI.
  GO TO ENO-OUTPUT. 00101400
OVER-FLW-STACK.
  PERFORM STACK-TB.
  MOVE 1 TO NO-OF-ELEMENTS (ICS).
  GC TO CONTINUE-OPO. 00101500
  00101600 00101700
CARRIAGE-CONTROL.
  IF NO-OF-ELEMENTS (ICS) NOT = 0 THEN PERFORM STACK-TB.
  MOVE 30 TO NO-OF-ELEMENTS (CSI).
  MCVE OUTSIGN TO SPECIAL-CONTROL ICS.
  IF OUTSIGN NOT = 72 THEN GC TO LOOP72.
  MCVE CNCHFR TO ELEMENT ICS, 31.
  PERFORM SHIFT-LEFT-CNE.
  IF STACK-INDICATOR = 2 THEN MOVE 5 TO STACK-INDICATOR.
  GC TO ENO-OUTPUT. 00102200
  00102300 00102400 00102500
LOOP72.
  IF (OUTSIGN NOT = 68) AND (OUTSIGN NOT = 71)
  THEN GC TO ENO-OUTPUT.
  MOVE FSTCHARS TO ELEMENT (CS), 3.
  PERFORM SHIFT-LEFT-CNE 2 TIMES.
  GC TO ENO-OUTPUT. 00102600
  00102700 00102800 00102900
STACK-CONTROL.
  IF OUTSIGN = 86 THEN GO TO UNIT-OF-MEASURE.
  IF OUTSIGN = 88 THEN GO TO BEGIN-TITLE.
  IF OUTSIGN = 89 THEN GO TO END-TITLE.
  IF OUTSIGN = 82 THEN GO TO ENO-HEADER.
  IF OUTSIGN = 81 THEN GO TO BEGIN-HEADER.
  IF OUTSIGN = 84 THEN IF STACK-INDICATOR = 2
    THEN MOVE 5 TO STACK-INDICATOR.
  GC TO ENO-OUTPUT. 00103100
  00103200 00103300 00103400
MCOE-CONTROL.
  IF OUTSIGN = 91 THEN GO TO SELF-TEST-OFF, ELSE
  IF OUTSIGN = 92 THEN GO TO SELF-TEST-ON, ELSE
  IF OLSIGN = 93 THEN PERFORM SET-TAB, ELSE
  IF OUTSIGN = 94 THEN PERFORM COMPLETE-CONTROL, ELSE
  IF OLSIGN = 95 THEN MCVE 1 TO POETRY-INDICATOR, ELSE
  IF OUTSIGN = 96 THEN MOVE 0 TO POETRY-INDICATOR, ELSE
  IF OUTSIGN = 97 THEN PERFORM COMPUTER-BRAILLE, ELSE
  IF OUTSIGN = 98 THEN GO TO WRAP-UP.
  GC TO ENO-OUTPUT. 00104100
  00104200 00104300 00104400
BEGIN-HEADER.
  00104500 00104600 00104700
  00104800 00104900
00105000
  00105100
  00105200
  00105300
  00105400
  00105500
  00105600
  00105700
  00105800
  00105900
  00106000
  00106100
GC TO LCOPI2.

TITLE2.
MCVE 1 TO CURRENT-TYPE.
MCVE 2 TO STACK-INDICATOR.
GC TO ENO-OUTPUT.

SELF-TEST-CFF.
ALTER S-T-B1 TO PROCEED TO S-T-NO.
MCVE 0 TO SELF-TEST.
GO TO ENO-OUTPUT.

SELF-TEST-CH.
ALTER S-T-B1 TO PROCEED TO S-T-YES1.
MCVE 1 TO SELF-TEST.
MCVE 6 TO CURRENT-TRANS.
MOVE I TO CURRENT-IN-TRANS.
MCVE 0 TO NO-OF-ELEMENTS-TRANS (1).
MOVE C 10 IN-CONSTRACT.
GC TO ENO-OUTPUT.

UNIT-OF-MEASURE.
MOVE I TO SPECIAL-CONTROLL (CS).
PERFORM TAIL-SWAP.
IF RLI = 'S' THEN PERFORM SHIFT-LEFT-ONE.
IF RLI NOT = 'S' THEN GO TO ENO-OUTPUT.
IF RCHAR (1) NOT = ' ' OR RCHAR (2) NOT = ' ' 
THEN PERFORM SHIFT-LEFT-CNE.
GO TO ENO-OUTPUT.

WRAP-UP.
MCVE 65 TO OUTSIGN.
PERFORM CARRIAGE.
CLOSE-FILES.
CLOSE SYSINPUT.
CLOSE SYSPRINT.
CLOSE SYSLRL.
CLOSE SYSPQ.
IF PUNCHED-OUTPUT NOT = 'N' THEN CLOSE PUNCH.
STOP RUN.

END-OUTPUT. EXIT.

CARRIAGE SECTION.
CARRIAGE-PARAGRAPH.
BREAK.
IF LNECOUNT < LPG 
THEN GO TO BREAKS.
MCVE SPACES TO OUT.
IF PROOF-OUT = 'P' THEN
WRITE OUTPLT-LINE AFTER C.
MCVE SPACES TO CUT-RPG.
IF RPC = 'R' THEN
WRITE OUTPUT-RPG AFTER 0.
MOVE SPACES TO CUT-RL.
IF BRAILLE = 'B' THEN
WRITE OUTPUT-BRL AFTER 0.
MOVE 0 TO LINECOUNT.
MCVE 0 TO CARDS-PRINTED.
PERFORM PAGE-HEAD.
BREAK.
IF OUTPTR = 1 THEN IF OUTSIGN = 65 OR OUTSIGN = 67
THEN GO TO BREAK12.
PERFORM CLT-OUTPUT.
BREAK12.
IF OUTSIGN = 65 OR OUTSIGN = 69 THEN MOVE 1 TO OUTPTR,
ELSE IF OLTSIGN = 67 THEN MOVE 3 TO OUTPTR,
ELSE IF OUTPTR = CRTL + 1 THEN MOVE 1 TO OUTPTR,
ELSE IF OUTPTR = 2 * CRTL THEN MOVE CRTL TO OUTPTR,
ELSE COMPLETE OUTPTR = OUTPTR - M.
ENO-CARRIAGE. EXIT.
CLEAR-BT SECTION.
CLEAR-BT-PARAGRAPH.
MOVE 2 TO STACK-INDICATOR.
MCVE 0 TO 1.
PERFORM CLEAR-STACKS.
MCVE PS TO CURRENT-S (CURRENT-TYPE).
ENO-CLEAR-BT. EXIT.
CLEAR-STACKS SECTION.
CLEAR-STACKS-PARAGRAPH.
    IF I = C THEN MCVE TS TO PS ELSE MOVE HS TO PS.
    MOVE 0 TO M.
    IF SPECIAL-CONTROL (PS) < 64
    THEN GO TO LOOP14.
    MCVE SPECIAL-CONTROL (PS) TO OUTSIGN.
    IF OUTSIGN NOT = 68 THEN GC TO LCP20.
    MCVE ELEMENT (PS, 3) TC TABULATE.
    PERFORM TABULATION.
    GC TO LCP22.
LOOP20.
    IF OUTSIGN NOT = 71 THEN GO TO LOOP30.
    IF OUTPTR = 1 THEN GC TO LOOP54.
    MOVE 65 TO OUTSIGN.
    PERFORM CARRIAGE.
LCP54.
    MCVE SPACES TO BRAILLE-LINES.
    MCVE SPACES TO PROCF-LINE.
    MOVE ZEROS TO CODE-LINE.
LCP27.
    IF M = ELEMENT (PS, 3) THEN GO TO LOOP22.
    COMPUTE OUTPTR = CRTL + 1.
    PERFORM CARRIAGE.
    AOO 1 TC M.
GO TO LCP27.

LCP26.
IF OILSIGN CNT = 72 THEN GO TO LCP26.
MOVE ELEMENT (PS, 3) TC 
IF TAB-TYPE (M) = 'R' THEN GO TO LCP73.
IF TAB-TYPE (M) = 'C' THEN GO TO LCP74.
MOVE TAB-COLUMN (M) TC TABULATE.
PERFORM TABULATION.
GO TO LCP22.

LCP72.
MOVE NEXT-STACK (PS) TC XX.
CCMpute TABULATE = TAB-COLUMN (M) - NO-OF-ELEMENTS (XX) + 1.
PERFORM TABULATION.
GO TO LCP22.

LCP74.
MOVE NEXT-STACK (PS) TO XX.
MOVE 1 TO XYZ.

LCP75.
IF ELEMENT (XX, XYZ) = 40 THEN GO TO LCP76.
ACO 1 TO XYZ.
IF XYZ CNT > NO-OF-ELEMENTS (XX) THEN GO TO LCP75.

LCP76.
CCMpute TABULATE = TAB-COLUMN (M) - XYZ + 1.
PERFORM TABULATION.
GO TO LCP22.

LCP26.
PERFORM CARRIAGE.
GO TO LCP22.

LCP14.
IF NO-OF-ELEMENTS (PS) = 0 THEN GO TO LCP22.
IF OUTPTR + NO-OF-ELEMENTS (PS) < OUTL + 2
THEN GO TC LCP5.
CCMpute OUTF PR = CTL + 1.
PERFORM CARRIAGE.
IF POETRY-INDICATOR = 1 THEN PERFORM MOVE-SPACE 2 TIMES.
IF HEAD-INO = 1 THEN PERFORM MOVE-SPACE 3 TIMES.

LCP5.
MOVE PS TC N.

LCP5R.
ACO 1 TC M.
PERFORM MOVE-ELEMENT.
IF M NOT = NO-OF-ELEMENTS (N) THEN GO TO LCP5R.
IF SPECIAL-CONTROL (N) = 1 THEN GO TO LCP22.
PERFORM MOVE-SPACE.

LCP22.
IF HS = TS THEN GO TO CCNE-CLEAR-STACKS.
IF I = 0 THEN PERFORM FREE-LAST-STACK.
ELSE PERFORM FREE-FAC-STACK.
GO TO CLEAR-STACKS-PARAGRAPH.
CCNE-CLEAR-STACKS.
MOVE 0 TO NO-OF-ELEMENTS (HS).
MOVE C TO SPECIAL-CCATFOL (HS).
END-CLEAR-STACKS. EXIT.

06119400
06119500
06119960
06119700
06119800
06119900
06120000
06120100
06120200
06120300
06120400
06120500
06120600
06120700
06120800
06120900
06121000
06121100
06121200
06121300
06121400
06121500
06121600
06121700
06121800
06121900
06122000
06122100
06122200
06122300
06122400
06122500
06122600
06122700
06122800
06122900
06123000
06123100
06123200
06123300
06123400
06123500
06123600
06123700
06124100
06124200
06124300
06124310
06124310
06124320
06124400
CLEAR-TR SECTION.
CLEAR-TR-PARAGRAPH.
  PCVE I TO 1.
  PERFORM CLEAR-STACKS.
  PCVE HE TO CURRENT-S (CURRENT-TYPE).
  END-CLEAR-TR. EXIT.

CCCOMPLETE-CCNTROL SECTION.
CCCOMPLETE-CCNTROL-PARAGRAPH.
  IF RLI = ' ' THEN GC TO ENO-CCCOMPLETE-CCNTROL.
  COMPUTE X = CNECHAR + 8 * RESTCHAR (1).
  IF X = 0 THEN GC TO LCOPE7a.
  ADD 1 TO NO-OF-ELEMENTS (CS).
  PCVE NO-OF-ELEMENTS (CS) TO NCS.
  MCVE X TO ELEMENT (CS, NCS).
LCOPE6a.
  PERFORM SHIFT-LEFT-CNE.
  PERFORM SHIFT-LEFT-CNE.
  GO TO COMPLETE-CCNTROL-PARAGRAPH.
LCOPE7a.
  PERFORM STACK-TB.
  GC TO LCOPE6a.
  END-CCCOMPLETE-CCNTROL. EXIT.

CCMP-BRAILLE SECTION.
CCMP-BRAILLE.
  IF RLI = ' ' THEN GC TO ENC-CCMP-BRAILLE.
  PCVE I TO LETTER.
LCOPE1a.
  IF SYMBOL (LETTER) = RLI THEN GC TO LCOPE6a.
  IF LETTER NOT < ALPHABET THEN GO TO CCMP-BRAILLE-ERROR.
  ACC I TO LETTER.
  GC TO LCOPE6a.
CCMP-BRAILLE-ERROR.
  IF RLI = RCG THEN GC TO ENC-CCMP-BRAILLE.
  PCVE 'NEW CHAR' TO CUT.
  PCVE RLI TO OLT-PLACE (12).
  WRITE OLTPUT-LINE AFTER 2.
  PCVE SPACES TO COUT.
  MOVE ** TO RLI.
  GC TO CCMP-BRAILLE.
LCOPE6a.
  ADD 1 TO NO-OF-ELEMENTS (CS).
  MOVE NO-OF-ELEMENTS (CS) TO NCS.
  IF CCMP8 (LETTER) = 0 THEN MCVE 64 TO ELEMENT (CS, NCS).
  ELSE MOVE CCMP8 (LETTER) TO ELEMENT (CS, NCS).
  PERFORM SHIFT-LEFT-CNE.
  GC TO CCMP-BRAILLE.
  END-CCMP-BRAILLE. EXIT.

FREE-HEAC-STACK SECTION.
FREE-HEAC-STACK-PARAGRAPH.
MOVE C TO NO-OF-ELEMENTS (H$).
MOVE 0 TO SPECIAL-CONTROL (H$).
IF NEXT-STACK (H$) = 0 THEN GO TO ENO-FREE-HEAD-STACK.
MOVE HS TO M$.
MCVE HEAD-FREE-STACK TO N$.
MOVE NEXT-STACK (M) TO HEAD-$S (CURRENT-TYPE)$.
MOVE HEAD-$S (CURRENT-TYPE)$ TO HS$.
MCVE $M$ TO HEAD-FREE-STACK.$.
MOVE 0 TO PREVIOUS-STACK (+S)$.
MCVE $N$ TO NEXT-STACK (M$).
ENO-FREE-HEAD-STACK. EXIT$.
FREE-LAST-STACK SECTION$.
FREE-LAST-STACK-PARAGRAPH$.
MOVE 0 TO NO-OF-ELEMENTS (TS)$.
MOVE 0 TO SPECIAL-CONTROL (TS)$.
IF PREVIOUS-STACK (TS) = 0 THEN GO TO ENO-LAST-STACK.
MOVE TS TO M$.
MCVE HEAD-FREE-STACK TO N$.
MOVE PREVIOUS-STACK (M) TO TAIL-$S (CURRENT-TYPE)$.
MOVE TAIL-$S (CURRENT-TYPE)$ TO TS$.
MOVE 0 TO NEXT-STACK (TS)$.
MOVE $N$ TO NEXT-STACK (M)$.
MCVE M$ TO HEAD-FREE-STACK.$.
ENO-LAST-STACK. EXIT$.
MCVE-ELEMENT SECTION$.
MOVE ELEMENT-PARAGRAPH$.
MOVE ELEMENT ($N$, $M$) TO X$.
MCVE-SPECIAL-SIGN$.
MCVE X TO CODEO-SIGN (OUTPTR)$.
IF X = 0 THEN MOVE 64 TO X$.
MCVE COST$-1$-$4$ (X) TO BRaille-SIGN ($1$, OUTPTR)$.
MOVE COST$-2$-$5$ (X) TO BRaille-SIGN ($2$, OUTPTR)$.
MCVE COST$-3$-$6$ (X) TO BRaille-SIGN ($3$, OUTPTR)$.
MOVE PROOF-CHARACTERS (X) TO PROOF-CHARS (OUTPTR)$.
ADD 1 TO OUTPTR$.
ENO-MCVE-ELEMENT. EXIT$.
MCVE-ERROR SECTION$.
MCVE-ERROR-PARAGRAPH$.
MOVE NO-OF-ELEMENTS (CS) TO N$.
IF N < 30 THEN ADD 1 TO N$.
MCVE 63 TO ELEMENT (CS, N)$.
MOVE N TO NO-OF-ELEMENTS (CS)$.
ENO-MCVE-ERROR. EXIT$.
MCVE-SPACE SECTION$.
MCVE-SPACE-PARAGRAPH$.
IF OUTPTR > OUTL THEN GO TO ENO-MOVE-SPACE$.
MOVE 00 TO CODEO-SIGN (OUTPTR)$.
MOVE COST$-1$-$4$ (64) TO BRaille-SIGN ($1$, OUTPTR)$.
MCVE COST$-2$-$5$ (64) TO BRaille-SIGN ($2$, OUTPTR)$.
MCVE COST$-3$-$6$ (64) TO BRaille-SIGN ($3$, OUTPTR)$.
MOVE PROOF-CHARACTERS (64) TO PROOF (CUTPTR).
AEO 1 TO OUTPTR.
END-PRINT-SPACE, EXIT.

CLT-OUTPUT SECTION.

CLT-CLT.
IF PROOF-CLT NOT = 'P' THEN GO TO BREAK21.
MOVE SPACES TO OUT.
IF CARDS-PRINTED = 0 THEN
WRITE OUTPLT-LINE AFTER 1.
MOVE 0 TO CARDS-PRINTED.
MOVE 1 TO NN.

BREAK9.
MOVE BRAILLE-TEXT (NN) TO OUT.
WRITE OUTPLT-LINE AFTER 1.
AEO 1 TO NN.
IF NN NOT > 3 THEN GO TO BREAK9.
MOVE PROOF-TEXT TO OUT.
WRITE OUTPLT-LINE AFTER 1.
MOVE 1 TO N.
IF PROOF (40) = 'ISK' THEN MOVE SC-ERROR-CT TO PROOF (40).

BREAK14.
MOVE CODEE-SIGN (N) TO PROOF (N).
AEO 1 TO N.
IF N NOT > OUTL THEN GO TO BREAK14.
MOVE PROOF-TEXT TO CLT.
WRITE OUTPLT-LINE AFTER 1.

BREAK21.
IF BRAILLE NOT = 'B' THEN GO TO BREAK31.
MOVE SPACES TO OUT-ERL.
WRITE OUTPLT-ERL AFTER 1.
MOVE 1 TO NN.
MOVE 1 TO N.
COMPUTE K = 3 * OLT.

BREAK10.
MOVE LINE-CHAR (N), N) TO CUT-PLACE-ERL (K).
AEO 1 TO N.
SUBTRACT 1 FROM K.
IF N NOT > OUTL * 3 THEN GO TO BREAK10.
WRITE OUTPLT-ERL AFTER 1.
AEO 1 TO N.
MOVE 1 TO N.
COMPUTE K = 3 * OUTL.
IF NN NOT > 3 THEN GC TO BREAK10.

BREAK21.
IF RPC NOT = 'R' THEN GO TO BREAK41.
MOVE SPACES TO OUT-RPC.
WRITE OUTPLT-RPC AFTER 1.
MOVE 1 TO NN.

BREAK23.
MOVE 1 TO K.
MOVE 1 TO N.
MOVE SPACE TO OUT-RPC-BEGIN.
BREAK32.
MCVE BRAILLE-SIGN (NN, N) TO OUT-RPQ-ONE (K).
ADD 1 TC h.
MOVE BRAILLE-SIGN (NN, N) TO CUT-RPQ-TWO (K).
ADD 1 TO N.
ADD 1 TO K.
WRITE CUTFUT-RPQ AFTER 1.
ADD 1 TO NN.
IF NN NOT > 3 THEN GO TO BREAK33.

BREAK41.
IF PUNCHED-OUTPUT = 'N' THEN GO TO BREAK51.
MCVE CODED-TEXT TO CODE-D-OUT.
WRITE CODED-OUTPLT AFTER POCKET-SELECT.

BREAK51.
MCVE SPACES TO PRCCF-TEXT.
MOVE ZEROS TO CODED-TEXT.
MCVE SPACES TO BRAILLE-TEXT (1).
MOVE SPACES TO BRAILLE-TEXT (2).
MCVE SPACES TO BRAILLE-TEXT (3).
IF OUTSIGN = 69 THEN MOVE LPG TO LINECOUNT
ELSE ADD 1 TO LINECOUNT.
END-CUT-OUT. EXIT.

PAGE-HEAD SECTION.
PAGE-HEAD-PARAGRAPH.
IF PAGINATION = 'N' THEN GO TO END-PAGE-HEAD.
COMPUTE N = (OUTL - TITLE-LENGTH + 2) / 2 + 1.
IF N < 1 THEN MOVE 1 TO N.
MOVE BRAILLE-LINES TC TITLE-LINES.
MCVE PROCFLINE TO TITLE-PROOF.
MCVE CODE-LINE TO TITLE-SIGN-CODE.
MOVE OUTPTR TO OUTPTRSV.
MCVE SPACES TO BRAILLE-LINES.
MOVE SPACES TO PRCCF-LINE.
MOVE ZEROS TO CODE-LINE.
MCVE N TO OUTPTR.
MCVE C TO XYZ.
MOVE HEAD-S (2) TC N.

LOOP42.
IF NO-OF-ELEMENTS (N) = 0 OR SPECIAL-CONTROL (N) NOT < 64
THEN GO TO LOOP42-CONT.

LOOP42A.
ADD 1 TO XYZ.
MCVE ELEMENT (N, XYZ) TC X.
PERFORM MCVE-SPECIAL-SIGN.
IF XYZ NOT = NO-OF-ELEMENTS (N) THEN GO TO LOOP42A.
IF SPECIAL-CONTROL (N) = 0 THEN PERFORM MOVE-SPACE.

LOOP42-CNT.
IF N = TAIL-S (2) THEN GO TO LOOP44.
MCVE 0 TO XYZ.
MCVE NEXT-STACK (N) TC N.
IF OUTPTR + NO-OF-ELEMENTS (N) > OUTL - 5 THEN GO TO LOOP44.
GC TO LCCP42.

LCOP44.
MCVE 1 TO N.
COMPUTE OUTPTR = CLTL - 4.

LCOP46.
IF CRT-PAGE-DIGIT (N) NOT = 0 THEN GC TO LCOP45.
AEO 1 TO N.
ADD 1 TO OUTPTR.
GC TO LCOP46.

LCOP45.
MCVE 6C TO X.
PERFORM MCVE-SPECIAL-SIGN.

LCOP47.
COMPUTE XYZ = CRT-PAGE-DIGIT (A) + 1.
MCVE DIGIT (XYZ) TO X.
PERFORM MOVE-SPECIAL-SIGN.
AEO 1 TO N.
IF N ACT > 4 THEN GC TO LCCP47.
ADD 1 TO CRT-PAGE.

LCOP43.
PERFORM CLT-CUTPUT.
MOVE CUTPTRSV TO OLTPT.
MCVE TITLE-LINES TO BRAILLE-LINES.
MOVE TITLE-PROOF TO PBCCF-LINE.
MOVE TITLE-SIGN-CCGE TO CODE-LINE.
END-PAGE-HEAD. EXIT.

PUSH SECTION.
STACK-BT.
IF HEAD-FREE-STACK = 0 THEN GO TO END-PUSH.
MOVE HEAD-FREE-STACK TO M.
MCVE NEXT-STACK (HEAD-FREE-STACK) TO HEAD-FREE-STACK.
MCVE F5 TO N.
MCVE C TO PREVIOUS-STACK (M).
MCVE M TO PREVIOUS-STACK (A).
MOVE M TO HEAD-S (CURRENT-TYPE).
MCVE M TO F5.
MOVE M TO NEXT-STACK (M).
MCVE M TO CURRENT-S (CURRENT-TYPE).
MOVE M TO CS.
END-PUSH. EXIT.

SET-TAB SECTION.
SET-TAB-PARAGRAP.
MCVE RCHAR (1) TO TAB-TYPE (ONECHAR).
MOVE SCNOCHARS TO TAB-COLUMN (CNECHAR).
PERFORM SHIFT-LEFT-CNE 4 TIMES.
END-SET-TAB. EXIT.

SHIFT-LEFT-CNE SECTION.
SHIFT-LEFT-CNE-PARAGRAP.
MOVE RRS TO TEMP.
MCVE TEMP TO RL9.
MCVE PREVIOUS-STACK (N) TO NEXT-STACK (M).
END-TAIL-SWAP. EXIT.
TRANS-TEST SECTION.
TRANS-TEST-PARAGRAPH.
  IF NO-OF-ELEMENTS (CS) NCT =
  NC-OF-ELEMENTS-TRANS (CURRENT-TRANS) THEN GO TO FAIL-TRANS.
  IF NC-OF-ELEMENTS (CS) = 0 THEN GO TO PASS-TRANS.
  MOVE C TO N.
LOOP3.
  ADC 1 TC N.
  IF ELEMENT (CS, N) NCT =
  ELEMENT-TRANS (CURRENT-TRANS, N)
  THEN GC TC FAIL-TRANS.
  IF N < NC-OF-ELEMENTS (CS)
  THEN GC TO LOOP3.
PASS-TRANS.
  MCVE O TO WORD-ERROR.
  GO TO END-TRANS-TEST.
FAIL-TRANS.
  MCVE 1 TC WORD-ERROR.
END-TRANS-TEST. EXIT.
READ-CARD SECTION.
READ-CARD-PARAGRAPH.
  READ SYINPUT, AT END GO TO FIN
  MCVE CARD-XX TO CARD-NC.
  IF CARD-NC NOT > PREV-CARD-NC THEN GO TO SEQ-ERROR.
  MOVE CARD-NC TO PREV-CARD-NC.
  IF ECF+C NOT = "E" THEN GO TO ENO-READ-CARD.
  GO TO PRINT-CARD.
SEQ-ERROR.
  MOVE 'Y' TO SEQ-ERR-IND.
  MOVE ZERO TO PREV-CARD-NC.
  MOVE SPACES TO OUT.
  WRITE OUTPUT-LINE AFTER ADVANCING 1.
PRINT-CARD.
  MOVE TEXT TO OUT.
  WRITE OUTPUT-LINE AFTER ADVANCING 1.
END-READ-CARD. EXIT.
/*LKE0 EXEC PGM=IEWL,PARM=(XREF,List,LET), X00000130
 * REGICH=96K
 * SYSLIB DD ONSNAME=SYS1,COBOLIB,DISP=SHR
 * SYSLIB DD ONSNAME=LOADLIB,DISP=OLD,DELETE,UNIT=SYSOA
 * SYSUTI DD UNIT=SYSCA,SPACE=(1024,(50,20))
 * SYSMLO DD OSA=G&GCGO(SYS3),DISP=(NEW,PASS),
 *         UNIT=SYSCA,SPACE=(CYL,(1),1)
 * SYSPRT DD SYSOUT=A,
 *         DGB=(BLKSIZE=121,LRECL=121,RECFP=FEM)
 * EXEC PGM=CSYS3
 * STEPLIB DD OSA=G&GCG,DISP=(SHR,PASS)
 * SYSOOUT DD SYSOUT=A.
<table>
<thead>
<tr>
<th>WORD</th>
<th>VALUE</th>
<th>PAGE 035</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNOWLEDGE</td>
<td>0599959999</td>
<td></td>
</tr>
<tr>
<td>LESS</td>
<td>4014999999</td>
<td></td>
</tr>
<tr>
<td>LETTER</td>
<td>0723955555</td>
<td></td>
</tr>
<tr>
<td>LIKE</td>
<td>0799999999</td>
<td></td>
</tr>
<tr>
<td>LITTLE</td>
<td>0707959999</td>
<td></td>
</tr>
<tr>
<td>LORO</td>
<td>16C7555555</td>
<td></td>
</tr>
<tr>
<td>MANY</td>
<td>5613999999</td>
<td></td>
</tr>
<tr>
<td>MENT</td>
<td>4830559999</td>
<td></td>
</tr>
<tr>
<td>MORE</td>
<td>1399999999</td>
<td></td>
</tr>
<tr>
<td>MOTHER</td>
<td>1613999999</td>
<td></td>
</tr>
<tr>
<td>MUCH</td>
<td>1333955555</td>
<td></td>
</tr>
<tr>
<td>MUST</td>
<td>1312995999</td>
<td></td>
</tr>
<tr>
<td>MYSELF</td>
<td>1361119999</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>1625955995</td>
<td></td>
</tr>
<tr>
<td>NECESSARY</td>
<td>2917099999</td>
<td></td>
</tr>
<tr>
<td>NEITHER</td>
<td>2917099999</td>
<td></td>
</tr>
<tr>
<td>NESS</td>
<td>4814999999</td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td>2999999999</td>
<td></td>
</tr>
<tr>
<td>O'CLOCK</td>
<td>2108055555</td>
<td></td>
</tr>
<tr>
<td>OF</td>
<td>5599999995</td>
<td></td>
</tr>
<tr>
<td>ONE</td>
<td>1621959999</td>
<td></td>
</tr>
<tr>
<td>ONSELF</td>
<td>1621199995</td>
<td></td>
</tr>
<tr>
<td>ONG</td>
<td>4827999999</td>
<td></td>
</tr>
<tr>
<td>OU</td>
<td>5199955666</td>
<td></td>
</tr>
<tr>
<td>OUGHT</td>
<td>1651959999</td>
<td></td>
</tr>
<tr>
<td>OUAO</td>
<td>4025999999</td>
<td></td>
</tr>
<tr>
<td>OUNT</td>
<td>4030959999</td>
<td></td>
</tr>
<tr>
<td>OURSELVES</td>
<td>5123914995</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>5199999995</td>
<td></td>
</tr>
<tr>
<td>OW</td>
<td>4299955555</td>
<td></td>
</tr>
<tr>
<td>PAID</td>
<td>1525999999</td>
<td></td>
</tr>
<tr>
<td>PART</td>
<td>1615999995</td>
<td></td>
</tr>
<tr>
<td>PEOPLE</td>
<td>1599999995</td>
<td></td>
</tr>
<tr>
<td>PERCEIVE</td>
<td>1559093999</td>
<td></td>
</tr>
<tr>
<td>PERCEIVING</td>
<td>1559039297</td>
<td></td>
</tr>
<tr>
<td>PERHAPS</td>
<td>1559199995</td>
<td></td>
</tr>
<tr>
<td>QUESTION</td>
<td>1631999999</td>
<td></td>
</tr>
<tr>
<td>QUICK</td>
<td>3105959995</td>
<td></td>
</tr>
<tr>
<td>QLITE</td>
<td>3199999995</td>
<td></td>
</tr>
<tr>
<td>RATHER</td>
<td>2399999999</td>
<td></td>
</tr>
<tr>
<td>RECEIVE</td>
<td>23C9399599</td>
<td></td>
</tr>
<tr>
<td>RECEIVING</td>
<td>2309392799</td>
<td></td>
</tr>
<tr>
<td>REJOICE</td>
<td>2326595555</td>
<td></td>
</tr>
<tr>
<td>REJOICING</td>
<td>2326502799</td>
<td></td>
</tr>
<tr>
<td>RIGHT</td>
<td>1623966599</td>
<td></td>
</tr>
<tr>
<td>SAID</td>
<td>1425555555</td>
<td></td>
</tr>
<tr>
<td>SH</td>
<td>4199999999</td>
<td></td>
</tr>
<tr>
<td>SHALL</td>
<td>4199555555</td>
<td></td>
</tr>
<tr>
<td>SHOULD</td>
<td>4125999999</td>
<td></td>
</tr>
<tr>
<td>SICN</td>
<td>4029959999</td>
<td></td>
</tr>
<tr>
<td>SO</td>
<td>1499955555</td>
<td></td>
</tr>
<tr>
<td>SCME</td>
<td>1614559999</td>
<td></td>
</tr>
<tr>
<td>SPIRIT</td>
<td>5614959999</td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>1295559999</td>
<td>00001480</td>
</tr>
<tr>
<td>STILL</td>
<td>1295555555</td>
<td>00001490</td>
</tr>
<tr>
<td>SUCH</td>
<td>1433999555</td>
<td>00001500</td>
</tr>
<tr>
<td>TH</td>
<td>5795595555</td>
<td>00001510</td>
</tr>
<tr>
<td>THAT</td>
<td>3099955555</td>
<td>00001520</td>
</tr>
<tr>
<td>THE</td>
<td>4695559595</td>
<td>00001530</td>
</tr>
<tr>
<td>THEIR</td>
<td>5645555555</td>
<td>00001540</td>
</tr>
<tr>
<td>THEMSELVES</td>
<td>4613391495</td>
<td>00001550</td>
</tr>
<tr>
<td>THERE</td>
<td>1645555555</td>
<td>00001560</td>
</tr>
<tr>
<td>THESE</td>
<td>2446955555</td>
<td>00001570</td>
</tr>
<tr>
<td>THIS</td>
<td>5799595595</td>
<td>00001580</td>
</tr>
<tr>
<td>THOSE</td>
<td>2457595955</td>
<td>00001590</td>
</tr>
<tr>
<td>THROUGH</td>
<td>1657959595</td>
<td>00001600</td>
</tr>
<tr>
<td>THYSELF</td>
<td>5751115555</td>
<td>00001610</td>
</tr>
<tr>
<td>TIME</td>
<td>1630595555</td>
<td>00001620</td>
</tr>
<tr>
<td>TICN</td>
<td>4629999599</td>
<td>00001630</td>
</tr>
<tr>
<td>TO</td>
<td>2295995555</td>
<td>00001640</td>
</tr>
<tr>
<td>TODAY</td>
<td>3025955555</td>
<td>00001650</td>
</tr>
<tr>
<td>TOGETHER</td>
<td>3027239995</td>
<td>00001660</td>
</tr>
<tr>
<td>TONIGHT</td>
<td>3013995595</td>
<td>00001670</td>
</tr>
<tr>
<td>UNDER</td>
<td>1633559555</td>
<td>00001680</td>
</tr>
<tr>
<td>UPON</td>
<td>2437555555</td>
<td>00001690</td>
</tr>
<tr>
<td>US</td>
<td>3799999999</td>
<td>00001700</td>
</tr>
<tr>
<td>VERY</td>
<td>3995955595</td>
<td>00001710</td>
</tr>
<tr>
<td>WAS</td>
<td>5299959999</td>
<td>00001720</td>
</tr>
<tr>
<td>WERE</td>
<td>5499959999</td>
<td>00001730</td>
</tr>
<tr>
<td>WH</td>
<td>4995959999</td>
<td>00001740</td>
</tr>
<tr>
<td>WHERE</td>
<td>1645999999</td>
<td>00001750</td>
</tr>
<tr>
<td>WHICH</td>
<td>4995955595</td>
<td>00001760</td>
</tr>
<tr>
<td>WHOSE</td>
<td>2445595555</td>
<td>00001770</td>
</tr>
<tr>
<td>WILL</td>
<td>5899959999</td>
<td>00001780</td>
</tr>
<tr>
<td>WITH</td>
<td>6295555595</td>
<td>00001790</td>
</tr>
<tr>
<td>WORD</td>
<td>2458959999</td>
<td>00001800</td>
</tr>
<tr>
<td>WORK</td>
<td>1658999999</td>
<td>00001810</td>
</tr>
<tr>
<td>WORLD</td>
<td>5658555555</td>
<td>00001820</td>
</tr>
<tr>
<td>WOULD</td>
<td>5825959999</td>
<td>00001830</td>
</tr>
<tr>
<td>YOU</td>
<td>6195595999</td>
<td>00001840</td>
</tr>
<tr>
<td>YOUNG</td>
<td>1661959595</td>
<td>00001850</td>
</tr>
<tr>
<td>YOUR</td>
<td>6123999999</td>
<td>00001860</td>
</tr>
<tr>
<td>YOURSELF</td>
<td>6123115595</td>
<td>00001870</td>
</tr>
<tr>
<td>YOURSELVES</td>
<td>6123391495</td>
<td>00001880</td>
</tr>
</tbody>
</table>

<p>| SPACE | 14 CO 64 | CC |
|       | 01 17 17 | CO |
| E     | 01 16 16 | CO |
| A     | 01 01 01 | CO |
| S     | 01 14 14 | CC |
| T     | 01 30 30 | CC |
| I     | 01 10 10 | 00 |
| O     | 01 21 21 | CC |
| L     | 01 07 07 | CC |
| R     | 01 23 23 | CC |
| N     | 01 29 29 | CC |
| C     | 01 09 09 | CO |</p>
<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>01 25 25</td>
<td>00</td>
</tr>
<tr>
<td>P</td>
<td>01 15 15</td>
<td>00</td>
</tr>
<tr>
<td>M</td>
<td>01 13 13</td>
<td>00</td>
</tr>
<tr>
<td>U</td>
<td>01 37 37</td>
<td>00</td>
</tr>
<tr>
<td>LOGICAL NOT</td>
<td>03 22 00</td>
<td>00</td>
</tr>
<tr>
<td>PERIODICAL</td>
<td>03 40 40</td>
<td>00</td>
</tr>
<tr>
<td>B</td>
<td>01 03 03</td>
<td>00</td>
</tr>
<tr>
<td>F</td>
<td>01 11 11</td>
<td>00</td>
</tr>
<tr>
<td>V</td>
<td>01 39 39</td>
<td>00</td>
</tr>
<tr>
<td>G</td>
<td>01 27 27</td>
<td>00</td>
</tr>
<tr>
<td>W</td>
<td>01 58 58</td>
<td>00</td>
</tr>
<tr>
<td>Y</td>
<td>01 61 61</td>
<td>00</td>
</tr>
<tr>
<td>H</td>
<td>01 19 19</td>
<td>00</td>
</tr>
<tr>
<td>K</td>
<td>01 05 05</td>
<td>00</td>
</tr>
<tr>
<td>CDMA</td>
<td>03 42 02</td>
<td>00</td>
</tr>
<tr>
<td>J</td>
<td>01 26 26</td>
<td>00</td>
</tr>
<tr>
<td>L</td>
<td>01 31 31</td>
<td>00</td>
</tr>
<tr>
<td>X</td>
<td>01 45 45</td>
<td>00</td>
</tr>
<tr>
<td>Z</td>
<td>01 53 53</td>
<td>00</td>
</tr>
<tr>
<td>COLON</td>
<td>03 45 18</td>
<td>00</td>
</tr>
<tr>
<td>HYPHEN</td>
<td>03 36 36</td>
<td>00</td>
</tr>
<tr>
<td>$</td>
<td>02 02 01</td>
<td>00</td>
</tr>
<tr>
<td>%</td>
<td>02 52 26</td>
<td>00</td>
</tr>
<tr>
<td>&amp;</td>
<td>02 38 19</td>
<td>00</td>
</tr>
<tr>
<td>#</td>
<td>02 22 11</td>
<td>00</td>
</tr>
<tr>
<td>^</td>
<td>02 6C 03</td>
<td>00</td>
</tr>
<tr>
<td>&lt;</td>
<td>02 34 17</td>
<td>00</td>
</tr>
<tr>
<td>&gt;</td>
<td>02 18 09</td>
<td>00</td>
</tr>
<tr>
<td>/</td>
<td>02 50 25</td>
<td>00</td>
</tr>
<tr>
<td>?</td>
<td>02 20 10</td>
<td>00</td>
</tr>
<tr>
<td>*</td>
<td>02 54 27</td>
<td>00</td>
</tr>
<tr>
<td>&quot;</td>
<td>03 04 04</td>
<td>00</td>
</tr>
<tr>
<td>DOLLAR SIGN</td>
<td>03 43 50</td>
<td>00</td>
</tr>
<tr>
<td>_</td>
<td>13 56 40</td>
<td>01</td>
</tr>
<tr>
<td>&amp;</td>
<td>03 47 47</td>
<td>00</td>
</tr>
<tr>
<td>&quot;</td>
<td>03 52 02</td>
<td>01</td>
</tr>
<tr>
<td>#</td>
<td>03 33 20</td>
<td>00</td>
</tr>
<tr>
<td>%</td>
<td>03 41 15</td>
<td>00</td>
</tr>
<tr>
<td>&lt;</td>
<td>03 35 54</td>
<td>00</td>
</tr>
<tr>
<td>&gt;</td>
<td>03 28 54</td>
<td>00</td>
</tr>
<tr>
<td>/</td>
<td>01 00 08</td>
<td>00</td>
</tr>
<tr>
<td>?</td>
<td>03 57 38</td>
<td>00</td>
</tr>
<tr>
<td>\</td>
<td>03 46 22</td>
<td>00</td>
</tr>
<tr>
<td>,</td>
<td>03 4A 06</td>
<td>00</td>
</tr>
<tr>
<td>(</td>
<td>03 62 54</td>
<td>01</td>
</tr>
<tr>
<td>)</td>
<td>03 55 54</td>
<td>00</td>
</tr>
<tr>
<td>#</td>
<td>03 60 60</td>
<td>00</td>
</tr>
<tr>
<td>=</td>
<td>01 63 48</td>
<td>00</td>
</tr>
<tr>
<td>ENO ALPHABET</td>
<td>59 99 64</td>
<td>00</td>
</tr>
<tr>
<td>IN</td>
<td>01 03 03</td>
<td>6429599</td>
</tr>
<tr>
<td>WAS</td>
<td>01 04 64</td>
<td>4529999</td>
</tr>
<tr>
<td>WERE</td>
<td>01 05 64</td>
<td>4555599</td>
</tr>
</tbody>
</table>
$CPB1  C1 04 64579555
$CS1   01  03  36369999
$#1    01  02  64725999
$DCT1  C1  C4  64540965
     12  02  40609999
EN     P  C6  C3  40172955
ENOUGH 1  C7  01  00189999
_IN1   01  03  40209999
_WAS1  C1  C4  40529955
_WERE1  01  03  40549999
_HIS1  01  04  40389955
_RE1   01  03  40065555
_I1    01  02  99999999
&##1  C8  C3  67389999
# dummy
C#1   C8  02  67389955
C#1   C5  02  67454099
C#1   02  01  67599955
*##1  10  01  38999955
*#1   01  01  52959999
%###1  03  01  20209999
1###1 03  01  18159999
1<###1 C3  01  32549559
1>###1 03  01  54049959
1     15  C1  64058899
1    08  03  08172999
11    03  08172355
11    03  08172599
CNG1  01  02  02199995
A#1   01  03  08012399
_/ENOUGH_/ 01  10  34999999
_/EN_/1  C6  34999999
_/BE_/1  01  06  66999999
_/BY_/1  01  06  52999999
_/WAS_/1 01  06  52999999
_/WERE_/1 C6  52999999
_/HIS_/1  C7  38999959
_/LETTER_/ 01  10  07239999
_/INTO_/1  01  10  20229955
_/TO_/1  01  04  22989959
_/SH_/1  01  06  41999999
_/1    21  02  99999999
END OF TABLE  99  C0  99999999
02
P  14030213
L  01131313
NSW   08
NIC   82
RAS-RRRRRR-RRRRRRRR
SRRS-SRRRRRRRRR-RRRRSSRS
---------------------
----------SR----------
----------SR----------
SRRS-SRRRRRRRRR-RRRRSSS
LAST SIGN TABLE CARD

PROOF
00100100
NO BRAILLE
00100200
NO FG
00100300
NO PUNCHEO OUTLIN
00100400
SIGNS/LINE 38
00100500
LINES/PAGE 09
00100600
PAGINATION 0001

$TLS - SAMPLE - OCTSYS - RUN $TE PG $P $P - THIS TEST HAS THREE PARTS: $P
(1) REALISTIC RUNNING TEXT; $P
(2) SPECIALLY CONCEIVED EXERCISES OF THE CONTROL SYMBOLS AND OTHER FEATURES; AND $P
(3) THE FIRST PAGE OF THE APPROXIMATELY 5000 PROBLEM WORDS FROM THE
TRANScribers' GUIDE, TRANSLATED AND AUTOMATICALLY FLAGGED, IF WRONG,
BY THE SELF-CHECKING FEATURE.

$TLS - BUSINESS - DATA - PROCESSING$TE
$PG $HCS - EMBOSSED IN BRAILLE $OE $L $HOS ON AN IBM SYSTEM/360
COMPUTER $HEE $CEAT THE ATLANTA BOARD OF EDUCATION $HOE $PG $P $P
-FRONT $COVER $P - WHAT IS DATA PROCESSING? $P - WHY HAS IT CAUSED
A VIRTUAL REVOLUTION IN THE ADMINISTRATION OF MODERN BUSINESS? $P
- WHAT ARE THE TWO MAJOR MODERN PROCESSING SYSTEMS? $P - WHAT
PROBLEMS DO BUSINESSES FACE IN INITIATING MODERN DATA PROCESSING INTO
THE BUSINESS COMPLEX? $P - BUSINESS - DATA - PROCESSING - SECON
- EDITION BY ELIAS - AMERICAN ANSWERS ALL THESE QUESTIONS SIMPLY AND
COMPLETELY. IF YOUR POSITION REQUIRES YOU TO KNOW THE FUNDAMENTALS OF
MODERN DATA PROCESSING (ITS PRINCIPLES AND ITS EQUIPMENT), THIS
UP-TO-DATE BOOK IS ESSENTIAL READING FOR YOU. IT OFFERS SO
COMPREHENSIVE A TREATMENT OF

THE FIELD THAT YOU CAN EASILY ACQUIRE A FULL NONTECHNICAL MASTERY OF
MODERN DATA PROCESSING PRINCIPLES AND METHODS, WITH NO PRIOR BACKGROUND

$TLS SAMPLE TITLE
$PG

THE 36 BOYS MEASURE OFF 24 INCHES AND 36 MILES WITH THEIR 12 SLIDE RULERS. THE 2 BOYS LIFTED 53 POUNDS 1 FOOT 13 METERS IN THE AIR.

$PG $STAB06 $SAMPLE $RUN $P = ALL OF THE SPECIAL SYMBOLS.

AS DESCRIBED IN _-_OOTSYS --II --USER'S --GUIDE < KEYPUNCH < INSTRUCTIONS $00391300

> ARE USED. THE SAID: "I SAID, "THE SIGN FOR =K IS GIVEN THE PROOF SYMBOL =K EVEN WHEN IT REPRESENTS THE WORD $*$GKNOWLEDGE*$.$R$"R"

-THE EDITOR MAY WISH TO USE THE TERMS /NATORST.

$TLS $TLE

(A) = "C" (I) = "THIS" "WHY NOT" (K)

(C) = "A" = "C" "H" = 0 = "K" = "K" (I) = "I"

$OCT1152536372527CO035210210C7012

$STAB14 0009210

$CPBEAST10L2RNCMPMU-.BFVG 0009220

$STB1R14 0009230

$STB2L05 0009240

$STB3D09 0009250

$#3 456.503 0009260

$#4 4560 0009270

$#2 TESTLEFT 0009280

$#1 TESTRIGHT 0009290

$STB4R16$STB5R15$STB6R14$#4TESTING

$SL14 0009300

$STLS SECOND SAMPLE TITLE$TLE

$FDSAMPLE $HEADG$HCE 0009320

$HD$S $SAMPLE $HEADING $SHOE

$SPYYS

THIS IS A TEST OF THE POETRY LINE INDENT. $L

EACH SENTENCE THAT EXTENDS INTO A NEW LINE SHOULD BE INDENTED TWO SPACES $0093500

* SL THE POETRY SYMBOLS TEST WILL FOLLOW THIS LINE. $L

POETRY SYMBOLS $FT $CS $SV $LV

$SPYYE

$TLS 3RD SAMPLE TITLE $TLE$PG$HOSTHIS BOOK WEIGHS 3 LBS. 11 OZ. $HDE

$HOST96 LB. TEST RUNS$HVE $TLS SAMPLE TITLE 9 MILLIMETERS LONG $TLE $PG

3 YD. 2 FT. 4 IN; 3 YD., 2 FT., 4 IN.

NCW, CANCEL THE TITLE: $TLS $TLE $PG

$HOSTHIS IS A SAMPLE TABLE$SHOE

$STB1L02 $STB2R15 $STB3O25 $STB4D32

$#1 ITEM $#2 MAN$R $#3 RATING $#4 PRICE

$#1 SOAP $#2 ACME $#3 42.3 $#4 5.22

$#1 SUOS $#2 MILLER $#3 100 $#4 6.89

$#1 QUACKS $#2 DUCKS $#3 0 $#4 1000000.1

$#1 OVERFLOWFIELD $#2 MAG $#3 9599.99 $#4 CHEAP

_/ABOUT_/ /_EE_/ /_KNOWLEDGE_/ /

$PG TRY COO INPUT SYMBOLS:

Q &

$PG

$TLS 5000 TYPICAL AND PROBLEM WORDS $TLE

$PG THE FOLLOWING ARE TYPICAL AND PROBLEM WORDS LISTED IN THE TRANSLATION RS' GUIDE. THE SELF-CHECKING FEATURE WILL CAUSE THOSE INCORRECTLY TRANSLATED BY OOTSYS TO BE FLAGGED. SOME CORRECTLY TRANSLATED WORDS MAY
ALSO BE FLAGGED: THESE SPURIOUS ERRORS ARE EXPLAINED IN THE USERS' GUIDE.

NOTE ALSO THAT THE =AB OF =AB INITIO, THE =AL OF =AL FINE, AND
THE =X OF =X-RAY SHOULD BE PRECEDED BY A LITERAL SIGN. THE SELF-CHECKIN
G FEATURE DOES NOT PRESENTLY CHECK FOR THIS.

SPG $SCON/3/3/3/3

AB|AR|E |ABAR|ICH |ABALNE |AB|ANO|ON|ED| |AB|SHOC|C|ED|DO
|E|EO| |ABE|ACY |ABE|E |ABE|EY |AB|B|OTT |AB|REV|I|A|TION |AB|CO|D|O|2|G
|O|IN|I|AL|LY |ABOCH|IN|O|LI|S |A|RE|A|M |ABEC|O|ARI|AN |AB|EO|I |AB|EO|CO|O|0
|R|O|E|E|N|E| |AB|ER|RI|AN|E| |ABE|Y|AN|E| |AB |IN|I|T|IO|N |AB|LE|E|BO|O|E|O|D|0

$SCOFF SPG

THIS IS THE END OF THE RUN.
### Right Context Table

<table>
<thead>
<tr>
<th>Non-Terminal</th>
<th>Input Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>14 03 02 13</td>
</tr>
<tr>
<td>L</td>
<td>01 13 13 13</td>
</tr>
</tbody>
</table>
## DECISION TABLE

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>02</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>03</td>
<td>G</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>04</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>05</td>
<td>G</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>06</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>07</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>08</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>09</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>12</td>
<td>G</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>G</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>G</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>G</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>G</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STATE VARIABLE</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>02</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>03</td>
<td>N</td>
<td>N</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>04</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>05</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>06</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>07</td>
<td>N</td>
<td>N</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>08</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STATE VARIABLE</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>---------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>INPUT CLASS</td>
<td>R</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>R</td>
</tr>
<tr>
<td>2</td>
<td>R</td>
<td>R</td>
<td>K</td>
<td>K</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>K</td>
<td>K</td>
<td>R</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>R</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>K</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>K</td>
<td>K</td>
<td>R</td>
</tr>
<tr>
<td>8</td>
<td>K</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>K</td>
<td>K</td>
<td>R</td>
</tr>
<tr>
<td>9</td>
<td>P</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>K</td>
<td>K</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>K</td>
<td>R</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>R</td>
<td>K</td>
<td>R</td>
<td>K</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>11</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>K</td>
<td>-</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>R</td>
<td>-</td>
<td>S</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>13</td>
<td>-</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>K</td>
<td>K</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>14</td>
<td>K</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>15</td>
<td>K</td>
<td>P</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>R</td>
<td>R</td>
<td>S</td>
<td>K</td>
<td>K</td>
<td>R</td>
</tr>
<tr>
<td>16</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>K</td>
<td>K</td>
<td>R</td>
<td>K</td>
<td>K</td>
<td>R</td>
</tr>
<tr>
<td>17</td>
<td>K</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>K</td>
<td>K</td>
<td>R</td>
<td>K</td>
<td>K</td>
<td>R</td>
</tr>
<tr>
<td>18</td>
<td>K</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>R</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>R</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>R</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>K</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>K</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>R</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>R</td>
<td>A</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

ON $A N$ $I R M$ $S Y S T E M$ $C F J$ $C O M P U T E R$

AT THE $A T L A N T A$ $B O A R D$ OF $E D U C A T I O N$ $S P$ $S P$ $S P$

WHAT IS DATA $P R O C E S S I N G$? $S P$ $S P$ $W H Y$ HAS IT CAUSED
INITIATING MODERN DATA PROCESSING INTO THE BUSINESS

- PLEX 7

45 36 15 07 17 45 38 00 00 00 00 00 00 00 00 00 00 00 00

SECOND EDITION BY ELIAS H. AWAD, ANSWERS ALL THESE QUESTIONS SIMPLY AND

BUSINESS DATA PROCESSING, SECOND

00 00 32 00 30 37 14 10 48 14 32 32 25 01 30 01 00 32 32 15 23 17 14 14 44 00 21 22 46 00 03 37 14 10 48 14

THE BOOK SIMPLY AND - PLEXELY, IF YOUR POSITION REQUIRES YOU TO KNOW THE FUNDAMENTALS OF

MODERN DATA PROCESSING (ITS PRINCIPLES AND ITS EQUIPMENT), THIS

Requires You To S K THE FUNDAMENTALS OF

UP-TO-DATE BOOK IS ESSENTIAL READING FOR YOU. IT OFFERS SO

MODERN DATA PROCESSING: ITS PRINCIPLES AND ITS

EQUIPMENT, UP-TO-DATE BOOK IS

COMPREHENSIVE A TREATMENT OF
THE FIELD THAT YOU CAN EASILY ACQUIRE A FULL NONTECHNICAL MASTERY OF

EASILY ACQUIRE A FULL NONTECHNICAL

MASTERY OF MODERN DATA PROCESSING PRINCIPLES

THE 36 BOYS MEASURED OFF 24 INCHES AND 36 MILES WITH THEIR 12 SLIDE

AND METHODS, WITH NO PRIOR BACKGROUND

RULES. THE 2 BOYS LIFTED 53 POUNDS 1 FOOT 13 METERS IN THE AIR.
THE FOLLOWING ARE ESSENTIAL AND PROGRAM 45 W.S. LISTED IN
RS GUIDE. THE SELF-CHECKING FEATURE WILL CAUSE THOSE INCORRECTLY

THE TRANSCRIBES GUIDE. THE SELF-CHECKING
RS GUIDE. THE SELF-CHECKING...

TRANSLATED BY OUTSYS TO BE FLAGGED. SOME CORRECTLY TRANSLATED WORDS MAY

FUTURE W... CAUSE THOSE INCORRECTLY

TRANSLATED... OUTSYS TO BE FLAGGED. ALSO BE FLAGGED; THESE SPURIOUS ERRORS ARE EXPLAINED IN THE USERS' GUIDE.

CORRECTLY TRANSLATED... 45 W.S. MAY AL.

NOTE ALSO THAT THE =AB OF =AB INITIO, THE =AL OF =AL FINE, AND

FLAGGED; 45 THE SPURIOUS ERRORS ARE EXPLAINED IN THE...

USERS' GUIDE. NOTE ALL THE =AB OF =AB

THE =X OF =X-RAY SHOULD BE PRECEDED BY A LITERAL SIGN. THE SELF-CHECKIN

INITIO, THE =AL OF =AL FINE. ANOTHER =X OF =X-RAY

G FEATURE DOES NOT PRESENTLY CHECK FOR THIS.
## APPENDIX III

### WORDS INCORRECTLY TRANSLATED IN 5808 PROBLEM WORDS

<table>
<thead>
<tr>
<th>DOTSYS III (TABLES 10/70)</th>
<th>TRANSCRIBERS' GUIDE</th>
<th>DOTSYS III (TABLES 10/70)</th>
<th>TRANSCRIBERS' GUIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>abecedarian</td>
<td>abecedarian</td>
<td>conger</td>
<td>conger</td>
</tr>
<tr>
<td>ablegate</td>
<td>ablegate</td>
<td>dar/edevil</td>
<td>daredevil</td>
</tr>
<tr>
<td>acreage</td>
<td>acreage</td>
<td>deaminate</td>
<td>deaminate</td>
</tr>
<tr>
<td>ACTH</td>
<td>ACTH</td>
<td>dedication</td>
<td>dedication</td>
</tr>
<tr>
<td>fine</td>
<td>fine</td>
<td>denudative</td>
<td>denudative</td>
</tr>
<tr>
<td>althorn</td>
<td>althorn</td>
<td>disacch/arine</td>
<td>disacch/arine</td>
</tr>
<tr>
<td>Antigone</td>
<td>Antigone</td>
<td>dis/ease</td>
<td>dis/ease</td>
</tr>
<tr>
<td>areaway</td>
<td>areaway</td>
<td>distinguie</td>
<td>distinguie</td>
</tr>
<tr>
<td>backword</td>
<td>backword</td>
<td>do (musical note)</td>
<td>do</td>
</tr>
<tr>
<td>bandog</td>
<td>bandog</td>
<td>Doolittle</td>
<td>Doolittle</td>
</tr>
<tr>
<td>Beelzebub</td>
<td>Beelzebub</td>
<td>dumbbell</td>
<td>dumbbell</td>
</tr>
<tr>
<td>Beguine</td>
<td>Beguine</td>
<td>ed/entulous</td>
<td>ed/entulous</td>
</tr>
<tr>
<td>benefic</td>
<td>benefic</td>
<td>Ein/thoven</td>
<td>Ein/thoven</td>
</tr>
<tr>
<td>Benees</td>
<td>Benees</td>
<td>Faenza</td>
<td>Faenza</td>
</tr>
<tr>
<td>Benét</td>
<td>Benét</td>
<td>Fer/inghee</td>
<td>Fer/inghee</td>
</tr>
<tr>
<td>Ber/ing</td>
<td>Bering</td>
<td>firedrake</td>
<td>firedrake</td>
</tr>
<tr>
<td>Bethesda</td>
<td>Bethesda</td>
<td>Fotheringham</td>
<td>Fotheringham</td>
</tr>
<tr>
<td>bezique</td>
<td>bezique</td>
<td>froghopper</td>
<td>froghopper</td>
</tr>
<tr>
<td>binucleate</td>
<td>binucleate</td>
<td>furlong</td>
<td>furlong</td>
</tr>
<tr>
<td>Blindheim</td>
<td>Blindheim</td>
<td>furth/er/er</td>
<td>furth/er/er</td>
</tr>
<tr>
<td>Boer</td>
<td>Boer</td>
<td>gallinipper</td>
<td>gallinipper</td>
</tr>
<tr>
<td>brougham</td>
<td>brougham</td>
<td>garderobe</td>
<td>garderobe</td>
</tr>
<tr>
<td>Bundestag</td>
<td>Bundestag</td>
<td>gastight</td>
<td>gastight</td>
</tr>
<tr>
<td>cadenced</td>
<td>cadenced</td>
<td>genitourin/ary</td>
<td>genitourin/ary</td>
</tr>
<tr>
<td>Caen</td>
<td>Caen</td>
<td>Gingold</td>
<td>Gingold</td>
</tr>
<tr>
<td>canzone</td>
<td>canzone</td>
<td>goatherd</td>
<td>goatherd</td>
</tr>
<tr>
<td>Castlera/gh</td>
<td>Castlera/gh</td>
<td>Goer/ing</td>
<td>Goering</td>
</tr>
<tr>
<td>castlery</td>
<td>castlery</td>
<td>gooseneck</td>
<td>gooseneck</td>
</tr>
<tr>
<td>cen/time</td>
<td>cen/time</td>
<td>Grantham</td>
<td>Grantham</td>
</tr>
<tr>
<td>chalone</td>
<td>chalone</td>
<td>Greatheart</td>
<td>Greatheart</td>
</tr>
<tr>
<td>Chisholm</td>
<td>Chisholm</td>
<td>Hadassah</td>
<td>Hadassah</td>
</tr>
<tr>
<td>cicerone</td>
<td>cicerone</td>
<td>Hadrian</td>
<td>Hadrian</td>
</tr>
<tr>
<td>conamed</td>
<td>conamed</td>
<td>Hapgood</td>
<td>Hapgood</td>
</tr>
<tr>
<td>conenose</td>
<td>conenose</td>
<td>hartshorn</td>
<td>hartshorn</td>
</tr>
<tr>
<td>congealed</td>
<td>congealed</td>
<td>heartsease</td>
<td>heartsease</td>
</tr>
<tr>
<td>congee</td>
<td>congee</td>
<td>hedger/ow</td>
<td>hedgerow</td>
</tr>
<tr>
<td>congen/er</td>
<td>congen/er</td>
<td>Here (goddess)</td>
<td>Here</td>
</tr>
<tr>
<td>congenial</td>
<td>congenial</td>
<td>Hergesheimer</td>
<td>Hergesheimer</td>
</tr>
<tr>
<td>congenital</td>
<td>congenital</td>
<td>Himalayas</td>
<td>Himalayas</td>
</tr>
<tr>
<td>DOTSYS III (TABLES 10/70)</td>
<td>TRANSCRIBERS' GUIDE</td>
<td>DOTSYS III (TABLES 10/70)</td>
<td>TRANSCRIBERS' GUIDE</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------</td>
<td>--------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Holin/sh/ed hornblende</td>
<td>Holinshed hornblende</td>
<td>pandemonism pedometer</td>
<td>pandemonism pedometer</td>
</tr>
<tr>
<td>hyaena</td>
<td>hyaena</td>
<td>Persephone</td>
<td>Persephone</td>
</tr>
<tr>
<td>in/en/arrable inglenook</td>
<td>in/en/narrable inglenook</td>
<td>persever/ance</td>
<td>pigheaded/ance</td>
</tr>
<tr>
<td>insofar/irone</td>
<td>insofar</td>
<td>pokeroor</td>
<td>pokeroor</td>
</tr>
<tr>
<td>Iredell</td>
<td>Iredell</td>
<td>potherb</td>
<td>potherb</td>
</tr>
<tr>
<td>isinglass</td>
<td>isinglass</td>
<td>pronephros</td>
<td>pronephros</td>
</tr>
<tr>
<td>jibboom</td>
<td>jibboom</td>
<td>pros and cons</td>
<td>pros and cons</td>
</tr>
<tr>
<td>krone</td>
<td>krone</td>
<td>protonema</td>
<td>protonema</td>
</tr>
<tr>
<td>Letter/er Letterman</td>
<td>Letter/er Letterman</td>
<td>rar/eripe</td>
<td>rar/eripe</td>
</tr>
<tr>
<td>Lever (bros.)</td>
<td>Lever</td>
<td>rawhide</td>
<td>rawhide</td>
</tr>
<tr>
<td>ling/eage (alignment) lineage</td>
<td>ling/erie</td>
<td>Reno</td>
<td>Reno</td>
</tr>
<tr>
<td>Littleton</td>
<td>Littleton</td>
<td>reredos</td>
<td>reredos</td>
</tr>
<tr>
<td>locoweed</td>
<td>locoweed</td>
<td>retroflex</td>
<td>retroflex</td>
</tr>
<tr>
<td>Loffler</td>
<td>Loffler</td>
<td>riboflavin</td>
<td>riboflavin</td>
</tr>
<tr>
<td>maenad</td>
<td>maenad</td>
<td>roped/ancer</td>
<td>ropedancer</td>
</tr>
<tr>
<td>Maugham</td>
<td>Maugham</td>
<td>saintonge</td>
<td>saintonge</td>
</tr>
<tr>
<td>mesitylene</td>
<td>mesitylene</td>
<td>sawhorse</td>
<td>sawhorse</td>
</tr>
<tr>
<td>Micronesian midwifery</td>
<td>Micronesian midwifery</td>
<td>Maugham</td>
<td>Maugham</td>
</tr>
<tr>
<td>minestrone</td>
<td>minestrone</td>
<td>sahmonge</td>
<td>sahmonge</td>
</tr>
<tr>
<td>mistitled</td>
<td>mistitled</td>
<td>sawhorse</td>
<td>sawhorse</td>
</tr>
<tr>
<td>More's (name) muraena</td>
<td>More's muraena</td>
<td>So (musical note)</td>
<td>So (musical note)</td>
</tr>
<tr>
<td>muraena</td>
<td>muraena</td>
<td>Some</td>
<td>Some</td>
</tr>
<tr>
<td>nathatch</td>
<td>nathatch</td>
<td>Soong</td>
<td>Soong</td>
</tr>
<tr>
<td>Oenone</td>
<td>Oenone</td>
<td>sou'ea/st/er</td>
<td>sou'ea/st/er</td>
</tr>
<tr>
<td>optime</td>
<td>optime</td>
<td>spathose</td>
<td>spathose</td>
</tr>
<tr>
<td>Osgood</td>
<td>Osgood</td>
<td>speakeasy</td>
<td>speakeasy</td>
</tr>
<tr>
<td>ou/ghtlins</td>
<td>ou/ghtlins</td>
<td>spikenard</td>
<td>spikenard</td>
</tr>
<tr>
<td>ou/thaul</td>
<td>ou/thaul</td>
<td>spumone</td>
<td>spumone</td>
</tr>
<tr>
<td>padrone</td>
<td>padrone</td>
<td>stagh/ound</td>
<td>stagh/ound</td>
</tr>
<tr>
<td>paleaceous</td>
<td>paleaceous</td>
<td>st/eroisomer</td>
<td>st/eroisomer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stirabout</td>
<td>stirabout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>string/endo</td>
<td>string/endo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>suede</td>
<td>suede</td>
</tr>
<tr>
<td></td>
<td></td>
<td>super/erogatory</td>
<td>super/erogatory</td>
</tr>
<tr>
<td>DOTSYS III (TABLES 10/70)</td>
<td>TRANSCRIBERS' GUIDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>suprar/enal</td>
<td>suprarenal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>swastikaed</td>
<td>swastikaed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>taenia</td>
<td>taenia</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tearoom</td>
<td>tearoom</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tearose</td>
<td>tearose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>teleran</td>
<td>teleran</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Theresa</td>
<td>Theresa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thiourea</td>
<td>thiourea</td>
<td></td>
<td></td>
</tr>
<tr>
<td>toadeater</td>
<td>toadeater</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tonelada</td>
<td>tonelada</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transmental</td>
<td>transmental</td>
<td></td>
<td></td>
</tr>
<tr>
<td>treenail</td>
<td>treenail</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trenal</td>
<td>trenal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>treponema</td>
<td>treponema</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tuberose</td>
<td>tuberose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tufthunter</td>
<td>tufthunter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tweedledee</td>
<td>tweedledee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tweedledum</td>
<td>tweedledum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'twouldn't</td>
<td>'twouldn't</td>
<td></td>
<td></td>
</tr>
<tr>
<td>undenomin/ational</td>
<td>undenomin/ational</td>
<td></td>
<td></td>
</tr>
<tr>
<td>underogating</td>
<td>underogating</td>
<td></td>
<td></td>
</tr>
<tr>
<td>us'n</td>
<td>us'n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vainglorious</td>
<td>vainglorious</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vandyke</td>
<td>Vandyke</td>
<td></td>
<td></td>
</tr>
<tr>
<td>violone</td>
<td>violone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wakerife</td>
<td>wakerife</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wenceslaus</td>
<td>Wenceslaus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>whaddaya</td>
<td>whaddaya</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wingate</td>
<td>Wingate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wired/ancer</td>
<td>wiredancer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wiredrawn</td>
<td>wiredrawn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wiseacre</td>
<td>wiseacre</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX IV

DEFINITION OF STATE VARIABLES AND INPUT CLASSES

State Variable
1  after the start of a number
2  after the start of a word
3  grade 1 translation
4  in a quotation
5  in italicized text
6  not at the start of a prefix or stem
7  part way through a word or phrase too long for one entry
8  just after a space (or A-J), following a number

Input Class
1  contractions always used in grade 2
2  digits
3  most punctuation
4  contractions used after the start of a word
5  $G$ (grade switch)
6  contractions used after the start of a word
7  isolated full-word contractions
8  $P''$ (start paragraph in quotation)
9  $P$ (start paragraph in italics)
10  " (left quote)
11  " (right quote)
12  ___ (begin italics)
Input Class 13 _ (last word of italics)
14 (space)
15 A to J or space occurring in a number
16 contractions always used in grade 2 containing terminal punctuation
17 prefix or first word of compound word
18 non-prefix beginning of word
19 first entry of double entry
20 second entry of double entry
21 "forced" contraction begin sign (/_)
22 units of measure and numbers following numbers
### APPENDIX V

#### SUMMARY OF SPECIAL SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Input Representation</th>
<th>Text Reference Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>capitalize letter</td>
<td>¬</td>
<td></td>
</tr>
<tr>
<td>capitalize word</td>
<td>¬¬</td>
<td></td>
</tr>
<tr>
<td>italicize word</td>
<td>_</td>
<td></td>
</tr>
<tr>
<td>italics start</td>
<td>_ _</td>
<td></td>
</tr>
<tr>
<td>left single quote</td>
<td>$'</td>
<td></td>
</tr>
<tr>
<td>right single quote</td>
<td>$'\text{R}'</td>
<td></td>
</tr>
<tr>
<td>left double quote</td>
<td>$&quot;</td>
<td></td>
</tr>
<tr>
<td>right double quote</td>
<td>$&quot;\text{R}'</td>
<td></td>
</tr>
<tr>
<td>accent marks</td>
<td>£</td>
<td></td>
</tr>
<tr>
<td>left bracket ([])</td>
<td>&lt;</td>
<td></td>
</tr>
<tr>
<td>right bracket ([])</td>
<td>&gt;</td>
<td></td>
</tr>
<tr>
<td>start paragraph</td>
<td>$\text{P}'</td>
<td></td>
</tr>
<tr>
<td>begin new line</td>
<td>$\text{L}'</td>
<td></td>
</tr>
<tr>
<td>begin new page</td>
<td>$\text{PG}'</td>
<td></td>
</tr>
<tr>
<td>skip to column nn</td>
<td>$\text{TABnn}'</td>
<td></td>
</tr>
<tr>
<td>letter sign</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>change grade</td>
<td>$\text{G}'</td>
<td></td>
</tr>
<tr>
<td>divide word</td>
<td>$\text{/}'</td>
<td></td>
</tr>
<tr>
<td>termination sign</td>
<td>$\text{T}'</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Input Representation</td>
<td>Text Reference Page</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>----------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>long vowel sign</td>
<td>$LV</td>
<td></td>
</tr>
<tr>
<td>start poetry</td>
<td>$PTYS</td>
<td></td>
</tr>
<tr>
<td>end poetry</td>
<td>$PTYE</td>
<td></td>
</tr>
<tr>
<td>start paragraph within quotation</td>
<td>$P&quot;</td>
<td></td>
</tr>
<tr>
<td>start title</td>
<td>$TLS</td>
<td></td>
</tr>
<tr>
<td>end title</td>
<td>$TLE</td>
<td></td>
</tr>
<tr>
<td>forced blank</td>
<td>$B</td>
<td></td>
</tr>
<tr>
<td>start heading</td>
<td>$HDS</td>
<td></td>
</tr>
<tr>
<td>end heading</td>
<td>$HDE</td>
<td></td>
</tr>
<tr>
<td>skip lines</td>
<td>$SLnn</td>
<td></td>
</tr>
<tr>
<td>short vowel sign</td>
<td>$SV</td>
<td></td>
</tr>
<tr>
<td>set tab t to col. nn</td>
<td>$STBt[L,R or D]nn</td>
<td></td>
</tr>
<tr>
<td>turn self-checking on</td>
<td>$SCON$/$/$/$/</td>
<td></td>
</tr>
<tr>
<td>turn self-checking off</td>
<td>$SCOFF</td>
<td></td>
</tr>
<tr>
<td>end of poetry foot sign</td>
<td>$FT</td>
<td></td>
</tr>
<tr>
<td>computer braille</td>
<td>$CPB</td>
<td></td>
</tr>
<tr>
<td>caesura sign</td>
<td>$CS</td>
<td></td>
</tr>
<tr>
<td>call (permanent) tab t</td>
<td>$#t</td>
<td></td>
</tr>
<tr>
<td>octal braille</td>
<td>$OCT</td>
<td></td>
</tr>
<tr>
<td>start forced contraction</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>end forced contraction</td>
<td>_/</td>
<td></td>
</tr>
</tbody>
</table>
### APPENDIX VI

**EQUIVALENT SIGNS, SYMBOLS AND CODES**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>.</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>0</td>
<td>c (θ)</td>
<td>5 (&quot;</td>
<td>=</td>
<td>.</td>
<td>456</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>:</td>
<td>.</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>C</td>
<td>E</td>
<td>D</td>
<td>CH * SH</td>
<td>% WH</td>
<td>: TH</td>
</tr>
<tr>
<td>01</td>
<td>09</td>
<td>17</td>
<td>25</td>
<td>33</td>
<td>41</td>
<td>49</td>
<td>57</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>2</td>
<td>J</td>
<td>1</td>
<td>I</td>
<td>:</td>
<td>3</td>
<td>J</td>
<td>EN</td>
</tr>
<tr>
<td>02</td>
<td>10</td>
<td>18</td>
<td>26</td>
<td>34</td>
<td>42</td>
<td>50</td>
<td>58</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>F</td>
<td>H</td>
<td>G</td>
<td>GL</td>
<td>ED</td>
<td>$</td>
</tr>
<tr>
<td>03</td>
<td>11</td>
<td>19</td>
<td>27</td>
<td>35</td>
<td>43</td>
<td>51</td>
<td>59</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>4</td>
<td>ST /</td>
<td>IN</td>
<td>9</td>
<td>AR</td>
<td>&gt;</td>
<td>-</td>
<td>ING (</td>
</tr>
<tr>
<td>04</td>
<td>12</td>
<td>20</td>
<td>28</td>
<td>36</td>
<td>44</td>
<td>52</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>K</td>
<td>M</td>
<td>O</td>
<td>N</td>
<td>U</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>05</td>
<td>13</td>
<td>21</td>
<td>29</td>
<td>37</td>
<td>45</td>
<td>53</td>
<td>61</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>S</td>
<td>TO</td>
<td>6</td>
<td>T</td>
<td>?</td>
<td>8</td>
</tr>
<tr>
<td>06</td>
<td>14</td>
<td>22</td>
<td>30</td>
<td>38</td>
<td>46</td>
<td>54</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>P</td>
<td>R</td>
<td>Q</td>
<td>V</td>
<td>AND</td>
<td>&amp;</td>
</tr>
<tr>
<td>07</td>
<td>15</td>
<td>23</td>
<td>31</td>
<td>39</td>
<td>47</td>
<td>55</td>
<td>63</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>first octal digit</th>
<th>second octal digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Braille sign</td>
<td>Proof character(s)</td>
</tr>
<tr>
<td>Computer braille graphic</td>
<td></td>
</tr>
<tr>
<td>Sign code</td>
<td>FORMAT</td>
</tr>
</tbody>
</table>

**FORMAT**

Included only where different from proof character. Parentheses around the computer braille graphic indicate that it is not implemented in the tables listed in Appendix I (and will be translated as a blank)
## APPENDIX VII

### TRANSLATOR - STACKER SIGN CODES

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>required blank</td>
</tr>
<tr>
<td>01-63</td>
<td>braille sign codes</td>
</tr>
<tr>
<td>64</td>
<td>end of braille word (blank or end of line)</td>
</tr>
<tr>
<td>65</td>
<td>new line</td>
</tr>
<tr>
<td>66</td>
<td>unused</td>
</tr>
<tr>
<td>67</td>
<td>paragraph start</td>
</tr>
<tr>
<td>68</td>
<td>one-time tab (skip to col. nn)</td>
</tr>
<tr>
<td>69</td>
<td>new page</td>
</tr>
<tr>
<td>70</td>
<td>unused</td>
</tr>
<tr>
<td>71</td>
<td>skip (multiple) lines</td>
</tr>
<tr>
<td>72</td>
<td>tab (skip according to permanent tab)</td>
</tr>
<tr>
<td>73-80</td>
<td>unused</td>
</tr>
<tr>
<td>81</td>
<td>start heading input</td>
</tr>
<tr>
<td>82</td>
<td>end heading input</td>
</tr>
<tr>
<td>83</td>
<td>unused</td>
</tr>
<tr>
<td>84</td>
<td>set continuous text stacking mode</td>
</tr>
<tr>
<td>85</td>
<td>idle (reserved for: reset continuous text stacking mode)</td>
</tr>
<tr>
<td>86</td>
<td>unit of measure</td>
</tr>
<tr>
<td>87</td>
<td>unused</td>
</tr>
<tr>
<td>88</td>
<td>start running title input</td>
</tr>
<tr>
<td>Code</td>
<td>Meaning</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>89</td>
<td>end of running title input</td>
</tr>
<tr>
<td>90</td>
<td>unused</td>
</tr>
<tr>
<td>91</td>
<td>self-checking mode off</td>
</tr>
<tr>
<td>92</td>
<td>self-checking mode on</td>
</tr>
<tr>
<td>93</td>
<td>set tab</td>
</tr>
<tr>
<td>94</td>
<td>start octal braille</td>
</tr>
<tr>
<td>95</td>
<td>start poetry mode</td>
</tr>
<tr>
<td>96</td>
<td>end poetry mode</td>
</tr>
<tr>
<td>97</td>
<td>start computer-braille</td>
</tr>
<tr>
<td>98</td>
<td>end or run</td>
</tr>
<tr>
<td>99</td>
<td>(filler in contraction table)</td>
</tr>
</tbody>
</table>
APPENDIX VIII

MISCELLANEOUS CODED VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>STACK-INDICATOR</td>
<td>2</td>
<td>Normal text; clear stack after each entry</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Continuous stacking of title or heading</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Continuous stacking of normal text</td>
</tr>
<tr>
<td>CURRENT-TYPE</td>
<td>1</td>
<td>Normal text or heading stack</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Title stack</td>
</tr>
</tbody>
</table>

Note: logical indicator variables are generally coded according to the convention 0 = "off" = "no" = "false," 1 = "on" = "yes" = "true."


DISTRIBUTION LIST

INTERNAL

C-01
C. W. Farr

D-06
P. R. Vance

D-07
J. H. Burrows
J. J. Croke

D-11
C. E. Duke
J. F. Jacobs

D-12
C. A. Zr'aket

D-63
R. S. Nielsen

D-72
T. L. Connors
C. G. Crothers
F. Engel
M. T. Gattozzi
W. R. Gerhart (5)
J. Mitchell
L. M. Thomas

D-73
W. Amory
N. W. Anschuetz
E. H. Bensley
J. A. Clapp
R. A. J. Gildea (20)
J. B. Glore
E. L. Lafferty (7)
J. F. Jacobs
J. K. Millen (20)
C. M. Sheehan
J. E. Sullivan (5)
N. B. Sutherland
E. W. Williamson

PROJECT

G. Dalrymple, M.I.T.
M. Leonard, M.I.T.
Prof. R. W. Mann, M.I.T.
V. A. Proscia, M.I.T. (50)

EXTERNAL

H. Bassler
Colonial Penn Insurance Co.
112 South 16th Street
Philadelphia, Pennsylvania 19102

Dr. M. P. Boyles (10)
Director, Computer-Braille Project
Instructional Services Center
Atlanta Public Schools
2930 Forrest Hill Drive, S. W.
Atlanta, Georgia 30315

L. L. Clark (2)
Director, IRIS
American Foundation for the Blind
15 West 16th Street
New York, New York 10011

E. L. Glaser
Computation Center
Case Western Reserve University
University Circle
Cleveland, Ohio 44106

Dr. C. E. Hallenbeck
Department of Psychology
University of Kansas
Lawrence, Kansas 66044

R. Haynes
American Printing House for the Blind
1839 Frankfort Avenue
Louisville, Kentucky 40206
EXTERNAL (Concluded)

Dr. K. R. Ingham
Room 20-B-207
Massachusetts Institute
of Technology
77 Massachusetts Avenue
Cambridge, Massachusetts 02139

R. E. LaGrone
IBM Corporation
Federal Systems Division
Department PC4, Room 2P25
18100 Frederick Pike
Gaithersburg, Maryland 20760

Dr. L. Leffler
Applied Mathematics Division
Argonne National Laboratories
9700 South Cass Avenue
Argonne, Illinois 60440

R. J. McNaughton
RCA Aerospace Systems Division
Burlington, Massachusetts 01801

Prof. A. Nemeth
Mathematics Department
University of Detroit
4001 W. McNichols
Detroit, Michigan 48821

Dr. B. Perella
Department of Defense
Fort George Meade, Maryland

A. Schack
Schack Associates
127 West 12th Street
New York, New York 10011

J. Siems
American Printing House for the Blind
1839 Frankfort Avenue
Louisville, Kentucky 40206

Dr. E. J. Waterhouse
Director
Perkins School for the Blind
175 North Beacon Street
Watertown, Massachusetts 02172

V. Zickle
American Printing House for the Blind
1839 Frankfort Avenue
Louisville, Kentucky 40206

FOREIGN

P. W. F. Coleman
4 George Street
Eastleight, Hants
S05 4 BU, United Kingdom

C. W. Garland
Royal National Institute for the Blind
224-6 Great Portland Street
London, W. 1, England

R. House
c/o Frank Giannone
1510 E. 11th Avenue
Vancouver 13, B.C.
Canada
DISTRIBUTION LIST (Concluded)

FOREIGN (Concluded)

Hans Karlgren
Research Group for Quantitative Linguistics
Sodermalmstorg 8,
11645 Stockholm
Sweden

Dr. G. Lamprecht
Wistfalische Wilhelms Universitat
44 Munster, Germany
Roxeler Strasse 64

B. Lindqvist
De Blindas Forening
Utvechlingsavdelningen
Gotlandsgatan 46
116 65 Stockholm
Sweden

J. Lindstrom
Handikappinstitutet
Pa Iach, 161 03 Bromma 3
Sweden

V. Mokleby
Husby OFF Skole for Blinde
Hovseterveien 3
Oslo 7
Norway

Prof. H. Werner
Institute fur Numerische und Instrumentelle Mathematik
Universitat Munster
Schlossplatz 5
44 Munster, West Germany

W. Sorke
Deutsche Blindenstudienanstalt
355 Marburg/Lahn
Am Schlag 8
West Germany