DOCUMENT RESUME

ED 074 152 TM 002 517

AUTHOR Koether, Mary E.; Coke, Esther U.
TITLE A Scheme for Text Analysis Using Fortran.
PUB DATE Feb 73
NOTE 27p.; Paper presented at annual meeting of the American Educational Research Association (New Orleans, Louisiana, February 25-March 1, 1973)

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS *Algorithms; Computational Linguistics; *Computer Programs; Grading; Language Patterns; *Programing Languages; Readability; Reading Speed; Speeches; *Structural Analysis; Tables (Data); Technical Reports; Test Bias; *Written Language

ABSTRACT

Using string-manipulation algorithms, FORTRAN computer programs were designed for analysis of written material. The programs measure length of a text and its complexity in terms of the average length of words and sentences, map the occurrences of keywords or phrases, calculate word frequency distribution and certain indicators of style. Trials of the programs, in studies of readability and reading rate, in aiding editors, in grading essays, and in identifying sources of response bias in multiple choice tests, demonstrate the potential applications of these algorithms in educational research and the usefulness of augmenting FORTRAN's computational facilities with character-processing capability. (Author)

A Scheme for Text Analysis Using Fortran

Mary E. Koether and Esther U. Coke
Bell Laboratories
Murray Hill, New Jersey

The title of this talk is "A Scheme for Text Analysis Using Fortran". If you have been drawn here by the error in the program that announces "A Scheme for TEST Analysis", let me encourage you to stay. You will hear about scoring tests as well as about analyzing texts.

Our scheme for text analysis with Fortran is based on the notion of English words as strings. We are indebted to a colleague, C. P. Imagna of Bell Laboratories, for suggesting this idea. I will explain the string concept in detail and then describe how we have used strings in writing Fortran programs to measure readability, develop editing aids and score tests.

The main point of my talk, however, is not to describe our programs, which were written to meet specific research needs. Rather, we believe that by using the string concept, any educational researcher who has access to Fortran can write his own programs tailored to his own text-analysis needs.

Let me begin by pointing out what we are not talking about. We are not describing a higher-level language, such as Snobol, designed specifically for character-manipulation. We are not talking about a package of programs,

such as the General Inquirer, designed to perform a specific kind of text-analysis. We are talking about adding an alphanumeric character-manipulation capability to Fortran, a language designed to perform numeric operations. We feel this adaptation is useful because Fortran is a widely-known programming language available at most computer installations.

The string concept is described in Fig. 1 of the handout. A string is a sequence of adjacent alphanumeric characters defined by three parts. The string's character-sequence is a subset of a collection of characters we call the character-collection. Notice that these characters include letters, numbers, punctuation marks and blanks. The position of the character-sequence in the character-collection is defined by a string's pointer and length. The pointer points to the character position in the collection of the first character of the character-sequence. The string's length equals the number of characters in the character-sequence.

In Fig. 1, the character-sequence, -mouse-, is defined by a pointer of 5 and length of 5. A pointer of 1 and length of 25 defines a character-sequence identical to the character-collection in the example. The character-collection remains the same regardless of changes in pointer and length values. Any subset of adjacent characters can be brought into focus by setting the pointer and length to

values that define the characters as a sequence. English words or nonsense syllables can be character-sequences. By varying the pointer and length values, a researcher may examine the same piece of text in a variety of ways.

Those of you who are familiar with Fortran may wonder how it is possible to manipulate a single character with standard Fortran. It isn't possible. Fortran's basic storage unit is the computer word. Therefore, Fortran's word-handling capabilities must be augmented with two character-manipulating subroutines. One of these routines stores a single character within a computer word; the other routine fetches a single character from within a computer word. These character-manipulation routines exist as system subroutines at many computer installations. If such ready-made subroutines are not available, they can be written by anyone familiar with assembly language coding.

Figures 2, 3 and 4 of the handout show how we have used the string concept in writing programs. The string is stored in a Fortran array, as shown in Fig. 2. The first computer word of the array holds the string's pointer; the second word holds the string's length and the remaining computer words of the array hold the string's character-collection. Different strings may be referred to by changing the values of the first two computer words of the array.

Figure 3 of the handout shows one basic operation of our text-analysis programs - segmenting a text string into words and sentences. A text is segmented by examining each character

in sequence to see if it is one of a special set of characters we call breaks and enders. The segments of text delimited by breaks and enders are called words and sentences. Example A uses standard punctuation marks as breaks and enders. In this case, the text is segmented into words and sentences as we normally think of them. However, other sets of breaks and enders may be used. Example C shows the segmentation of the text into words and sentences that do not conform to our normal usage. If you attempt this scan, you will appreciate the computer's advantage over man in segmenting texts according to special rules.

Each word or sentence segment has a pointer and length that defines it as a character-sequence within the text string. The values of the pointers and lengths can be saved for later use. These values can act as entries to the words and sentences stored within the character-collection of a text string.

That a single character-collection can serve as the source for many different character-sequences is an important feature of the string concept. The use of this feature in programming is illustrated in Fig. 4.

The purpose of the operation described in Fig. 4 is to store one copy of each word type found in the input records of a text. In the example, an input record has been stored as a string in the array, ITXT. The array, IWORD, at the top of the page holds one copy of each word type. Each time

a new word type is found its character-sequence is moved from ITXT to the end of the IWORD character-collection. The pointer to the beginning of the new word's character-sequence in IWORD and its length are stored in IREF.

In the figure, the flow of the program has reached the decision about the word -the- in ITXT. Is there a copy of this word in IWORD? This decision is made by comparing the string in ITXT with each string whose pointer and length are in IREF. The character-collection of IWORD is the character-collection for all the strings in IREF.

Two strings are identical if their character-sequences match. The determination of identity normally requires a character-by-character comparison of the strings. However, certain programming strategies may shorten search time. For example, there is no need to compare strings of different lengths.

The location in IREF of a word's pointer and length may serve as the word's numeric identifier in programming. For example, in the lower right-hand corner of Fig. 4, word numbers have been stored in ISEN to represent the sequence of words in ITXT. The pattern of numbers in ISEN might be used to recover recurrent phrases in a text. Or the numeric identifiers of the words could be used in making frequency counts of the different word types in a text.

Since each word type's character-sequence can be referenced through its pointer and length, a word's numeric

identifier also serves as an entry for retrieving the word's characters. These can be used in printing the word.

The programs that we have written to perform the operations of string storage, assignment of numeric identifiers and string retrieval are summarized in Table 1 of the handout. These utility subroutines have been used to write a number of general purpose text-analysis programs described in Table 2. Let me reemphasize that our text-analysis programs are only examples. Different research needs would and should produce different types of programs.

I would now like to illustrate some of the uses of the EZIER programs described in Table 2. These illustrations are intended only to suggest the range of analyses that can be undertaken with programs using the string concept.

The LETVOW program describes a text in terms of its overall length and its readability. The program estimates the number of syllables in a passage from a count of the number of vowels. The readability indices that the program calculates are mean sentence length in words and mean word length in syllables. While these measures can be made by hand, we have found that LETVOW does them with greater ease and with much greater accuracy when many long texts must be processed.

Figure 5 of the handout illustrates one way that LETVOW's analysis of individual sentences could be used to produce more readable texts. The idea behind the plots shown in Fig. 5 is that writers might communicate information more effectively if they avoided the use of long words in long

sentences. The plots show the joint distribution of sentence length in words and average word length in syllables for all sentences in two different texts. The numbers by each point are sentence reference numbers from our LOG program. This program prints a text with each sentence numbered. The lines dividing the plots into quadrants are used to define long sentences and long words. The definition of quadrants could be based on knowledge about readability. This kind of display could be used by an editor to locate sentences in the LOG printout that are too long or use lengthy words. Sentences lying in the upper right-hand quadrants should be especially good candidates for rewriting. Whether this editing procedure would produce more comprehensible texts has yet to be explored.

The FINDR program has also been generally useful. For example, we explored the possibility that certain words are diagnostic of the knowledge state of a writer. Hiller has suggested that words and phrases denoting vagueness would be good barometers of knowledge. We looked at short essays describing all that the writers could remember about material they just read. We found that Hiller's vagueness terms occurred rarely if at all in these short essays. Therefore, we selected words we felt to be indicative of vagueness from the 100 most frequently occurring words in American English. We found that the words, it and they, were used significantly more often in essays given a low score by a human scorer.

Since these essays had been scored with a checklist, we also tried scoring the essays by selecting a set of key words from the checklist. With our KEYS program we found the frequency with which checklist keywords occurred in the essays. As Hiller has just reported, keyword frequencies were highly correlated with scores assigned by the human scorer.

We have used our phrase-building program, FRS, to look for sources of response bias in multiple-choice tests. We looked at multiple-choice tests of comprehension associated with reading selections from a speed-reading course. We wanted to see if the words and phrases from the tests' correct alternatives occurred more often in the reading selections than the phrases from the error alternatives. If such were the case, a reader might choose correct alternatives on a multiple-choice test because these answers were more familiar and not because the reader understood the content of the passage.

With the FRS program we searched through each reading selection for the alternative answers in that selection's test. We used entire alternatives and components of the alternatives as phrases. This is illustrated in Fig. 6. The results of this analysis are shown in Table 3. They suggest that students saw proportionally more of the correct alternatives' phrases in the post-lesson reading selection than in the pre-lesson selection. This imbalance might predispose a student to perform better on the post-lesson test than on the pre-lesson test.

Finally, we have found our utility subroutines useful in writing programs to tabulate data. Figure 7 illustrates the tabulation of responses on a cloze test. These responses were packed onto cards for input to the utility subroutines, BRKRTN and WRDID, which separated and classified the responses. The output shown in Fig. 7 was used to select correct responses. These correct response types served as input to another program that scored the responses of the individual subjects. Programming of this type has made it possible to score large numbers of cloze tests quickly and reasonably inexpensively.

Our program applications have demonstrated to us that Fortran can handle alphanumeric materials efficiently and effectively. We also found it particularly useful to have Fortran's facility in numerical computation available as texts were being analyzed.

Our programs represent only one approach to text-analysis. These programs took the English word as their basic unit of analysis. Other research might focus on smaller or larger segments of text. The concept of a string would be just as effective for these sorts of analyses. The string concept's generality makes it potentially useful in many areas of educational research.

AERA ANNUAL MEETING

FEBRUARY 28, 1973

SESSION 29.22

A SCHEME FOR TEXT ANALYSIS USING FORTRAN

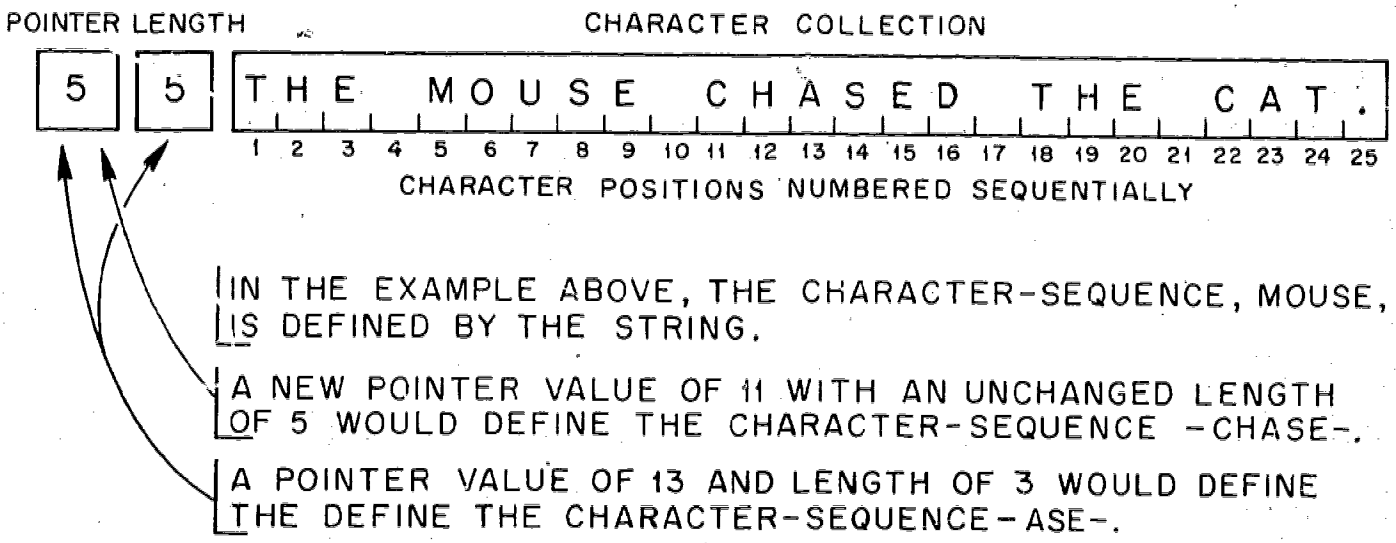
MARY E. KOETHER AND ESTHER U. COKE
BELL LABORATORIES
MURRAY HILL, NEW JERSEY

A STRING DEFINES A CHARACTER-SEQUENCE AND HAS THREE PARTS:

(1) A POINTER TO THE CHARACTER-POSITION OF THE FIRST CHARACTER OF A CHARACTER-SEQUENCE;

(2) THE LENGTH OF THE CHARACTER-SEQUENCE;

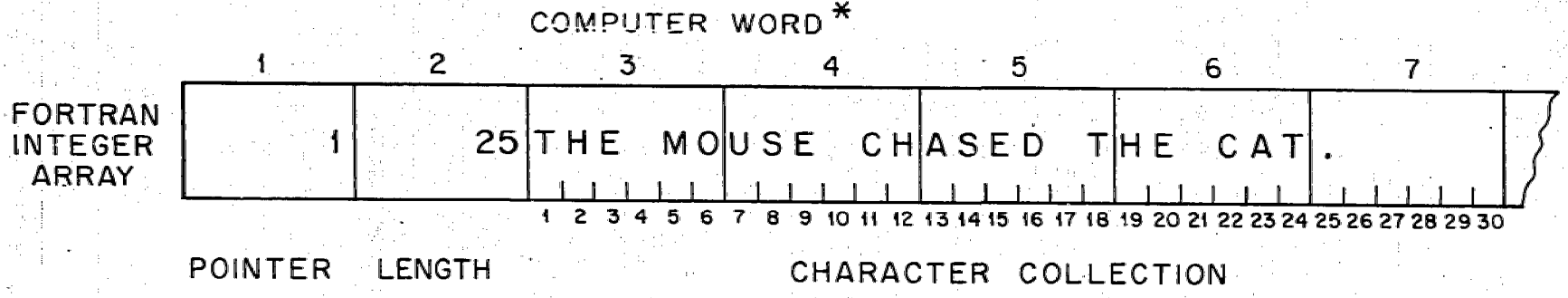
(3) A COLLECTION OF CHARACTERS OF WHICH THE CHARACTER-SEQUENCE IS A SUBSET.



CHANGES IN THE VALUE OF THE POINTER OR LENGTH DO NOT AFFECT THE CHARACTER-COLLECTION. IT REMAINS THE SAME.

FIG. 1
DIAGRAM OF A STRING

KOETHER & COKE
AERA, 1973



A STRING CAN BE STORED AS A FORTRAN INTEGER ARRAY WHOSE FIRST WORD CONTAINS THE STRING'S POINTER AND WHOSE SECOND WORD CONTAINS THE STRING'S LENGTH. THE REMAINDER OF THE ARRAY HOLDS THE CHARACTER-COLLECTION WITH THE FIRST CHARACTER-POSITION OF THE COLLECTION DEFINED AS THE FIRST CHARACTER-POSITION OF THE THIRD COMPUTER WORD OF THE ARRAY.

*COMPUTER WORD IS THE NAME OF THE BASIC UNIT FOR STORAGE OF INFORMATION IN A COMPUTER. THE NUMBER OF ALPHANUMERIC CHARACTERS STORED IN A COMPUTER WORD VARIES FROM COMPUTER TO COMPUTER. IN THIS EXAMPLE, SIX CHARACTERS ARE STORED IN EACH COMPUTER WORD.

FIG. 2
STRING STORAGE FOR COMPUTER PROGRAMMING

DEFINITIONS

A BREAK CHARACTER CAN BE ANY CHARACTER. THE SMALLEST USER-DEFINED UNIT, A WORD, IS THE LONGEST SEQUENCE OF CHARACTERS OCCURRING BETWEEN TWO BREAK CHARACTERS OR BETWEEN THE BEGINNING OF AN INPUT RECORD AND A BREAK CHARACTER.

AN ENDER CHARACTER CAN BE ANY CHARACTER. THE USER-DEFINED UNIT, A SENTENCE, IS THE LONGEST SEQUENCE OF CHARACTERS OCCURRING BETWEEN TWO ENDER CHARACTERS OR BETWEEN THE BEGINNING OF AN INPUT RECORD AND AN ENDER.

NOTE: ⊖ DENOTES A BLANK.

SAMPLE TEXT

FIRST,REMEMBER THAT YOU MAY CHOOSE BREAKS AND ENDERS.DO YOU UNDERSTAND?

EXAMPLE A

SEGMENTATION OF SAMPLE TEXT WITH THE TERMS WORD AND SENTENCE CORRESPONDING TO THEIR NORMAL ENGLISH USAGE.

BREAKS	SENTENCES	WORDS
⊖,:;	1	FIRST
		REMEMBER
ENDERS		THAT
.!?		YOU
		MAY
		CHOOSE
		BREAKS
		AND
		ENDERS
	2	DO
		YOU
		UNDERSTAND

EXAMPLE B

SEGMENTATION OF SAMPLE TEXT USING ONLY ONE BREAK CHARACTER AND ONE ENDER:

BREAKS	SENTENCES	WORDS
⊖	1	FIRST,REMEMBER
		THAT
ENDERS		YOU
		MAY
?		CHOOSE
		BREAKS
		AND
		ENDERS.DO
		YOU
		UNDERSTAND

FIG. 3

THE USE OF BREAK CHARACTERS AND ENDER CHARACTERS TO SE

INED UNIT.
N TWO BREAK
REAK CHARACTER.

T, A SENTENCE.
DER CHARACTERS

DERSTAND?

KAMPLE B
F SAMPLE TEXT USING ONLY
ACTER AND ONE ENDER.

SENTENCES	WORDS
1	FIRST, REMEMBER
	THAT
	YOU
	MAY
	CHOOSE
	BREAKS
	AND
	ENDERS. DO
	YOU
	UNDERSTAND

FIG. 3

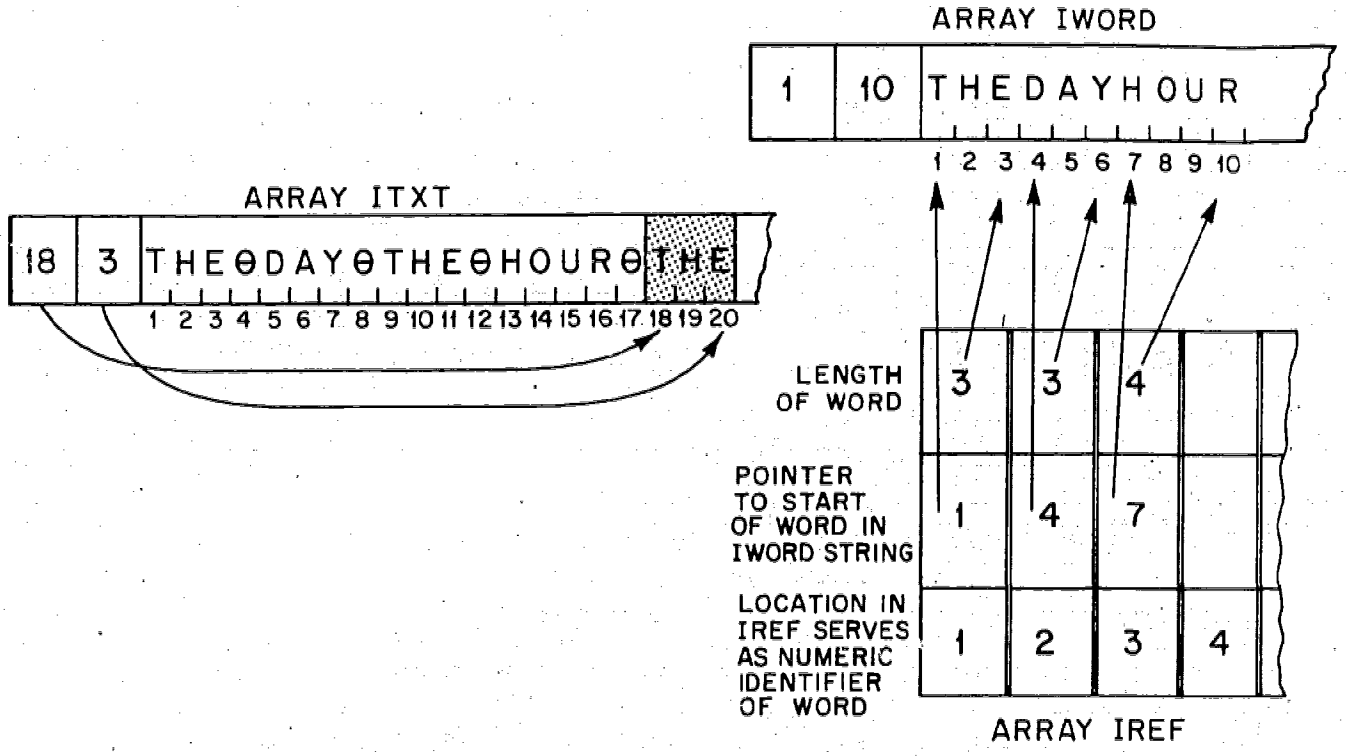
AND ENDER CHARACTERS TO S

EXAMPLE C

SEGMENTATION OF SAMPLE TEXT USING
VOWELS AS BREAK CHARACTERS AND
D AS THE SINGLE ENDER.

BREAKS	SENTENCES	WORDS
AEIOUY	1	F
		RST,R
		M
ENDERS		MB
		R~~Θ~~TH
D		T~~Θ~~
		~~Θ~~M
		~~Θ~~CH
		S
		~~Θ~~BR
		KS~~Θ~~
		N
	2	~~Θ~~
		N
	3	RS.
	4	~~Θ~~
		~~Θ~~
		N
	5	RST
		N
	---	---
	6	?

SENTENCE 6 IS UNDEFINED. THERE IS NO ENDER
CHARACTER DEFINING THE END OF 6.



START

DEFINE AS A STRING THE TARGET WORD TO BE STORED.

IN THIS EXAMPLE, -THE- IS THE WORD TO BE STORED. IT HAS BEEN DEFINED AS A STRING IN THE ARRAY ITXT.

DETERMINE WHETHER THE TARGET WORD HAS ALREADY BEEN STORED.

IN THE EXAMPLE, THE WORDS FOUND IN ITXT ARE STORED IN IWORD. NOTE THAT A WORD THAT OCCURS MORE THAN ONCE IN ITXT IS STORED ONLY ONCE IN IWORD. EACH WORD TYPE IN IWORD IS DEFINED AS A STRING WHOSE POINTER AND LENGTH ARE STORED IN IREF. THE LOCATION IN IREF OF A WORD'S POINTER AND LENGTH IS ITS NUMERIC IDENTIFIER. THIS NUMBER CAN BE USED INSTEAD OF THE WORD IN FURTHER PROGRAMMING.

FIG. 4

WORD STORAGE USING STRINGS AND N

KOETHER & COKE
AERA, 1973

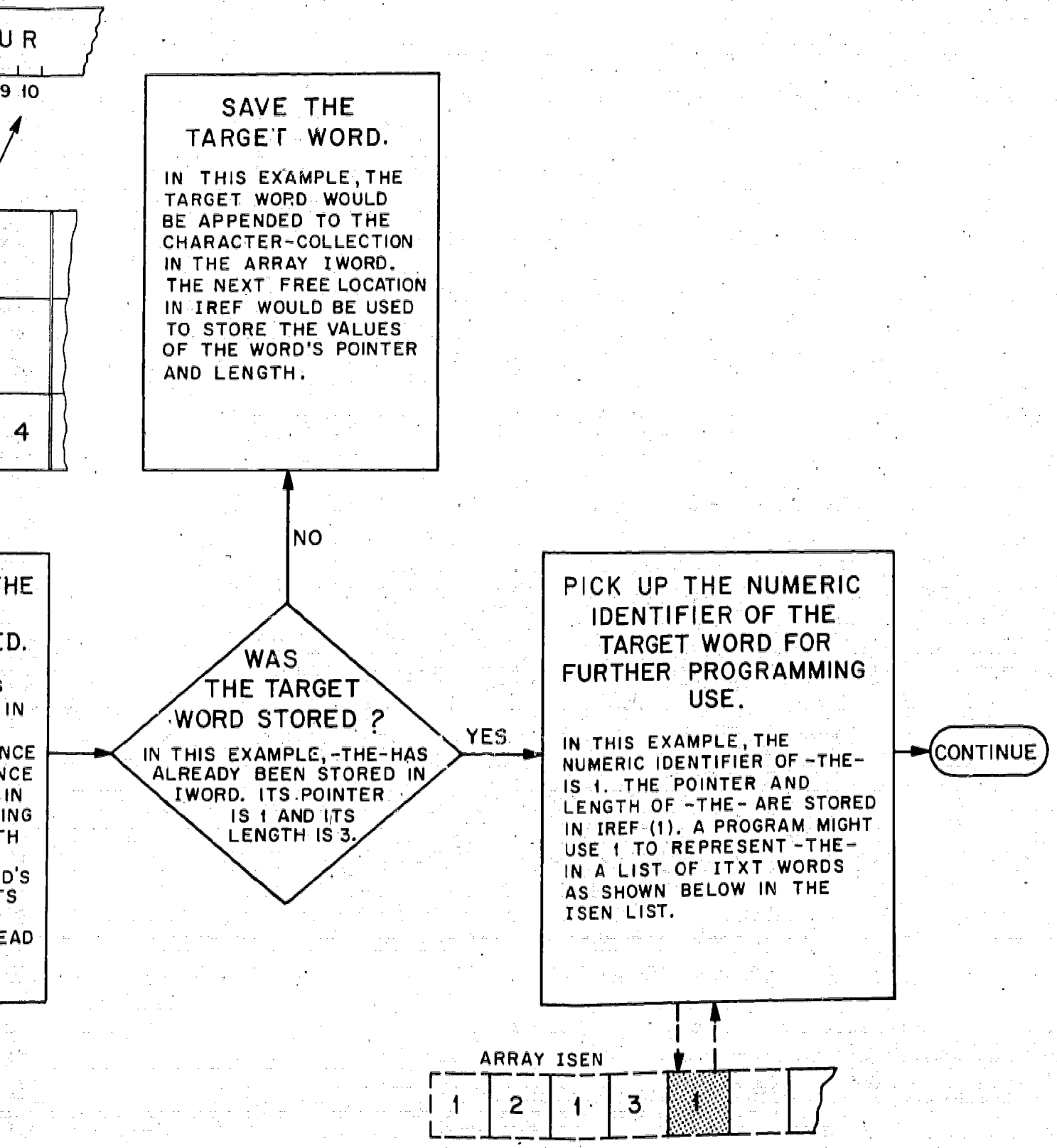


FIG 4

RINGS AND NUMERIC IDENTIFIERS

Table 1

A listing of EZIER utility subroutines with brief descriptions of their functions.

PROGRAM NUMBER	PROGRAM NAME	PROGRAM FUNCTIONS
8	LININ	Reads in text records and converts them to strings.
9	BRKRTN	Searches a string for a specific set of breaks and/or enders.
10	NUMVOW (NUMV)	Counts the number of vowels in a string.
11	NCHS	Counts the number of non-blank characters in an array.
12	KOMST (KOMA)	Compares two strings to determine if their character-sequences are identical.
13	WRDID	1) If a target string has not been encountered previously, the string is stored and assigned a unique identification number. 2) If a target string has already been encountered, the string's identification number is retrieved.
14	IDRTN	Retrieves the identification number assigned to a string by WRDID.
15	ICLAS	Classifies a string on the basis of its length.
16	OTSTR	Prepares a string for multiple-line printing so that words will not be continued from one printed line to the next.
17	PREPWD (PREPWN)	1) Calculates the number of six-character computer-words taken up by the character-sequence of a string. 2) Appends blanks to the end of a character-sequence if the end does not completely fill a computer word.

Table 1.1

PROGRAM NUMBER	PROGRAM NAME	PROGRAM FUNCTIONS
21	FNDBK1	Searches a string for one of a set of target characters stored in another string.
22	MOVST	Moves the character-sequence of one string to the beginning of the character-collection of another string.
23	APDST	Appends the character-sequence of one string to the end of the character-sequence of another string.
24	INSERT (STRING)	Inserts a set of characters from an array into a string.

Table 2

A listing of the EZIER text-analysis programs with brief descriptions of their functions.

PROGRAM NUMBER	PROGRAM NAME	PROGRAM FUNCTIONS
1.	LETVOW	1. Counts the number of letters, vowels and syllables in the text and in each sentence; 2. Calculates the mean sentence length in words and the mean word length in syllables for the text and the mean word length for each sentence of the text.
2.	FINDR	1. Calculates the frequency distribution of all word types in a text; 2. Lists word types alphabetically; 3. Calculates type-token ratio.
3.	KEYS	1. Searches text for all occurrences of words specified by the user; 2. Lists all the words found and their frequency of occurrence.
4.	FRS	1. Searches text for all occurrences of phrases specified by the user; 2. Lists all the phrases found and their frequency of occurrence.
5.	LENSEN	1. Calculates the distribution of word lengths in letters for each sentence and the text; 2. Calculates frequency distribution of sentence lengths in words.
6.	KYSDAT	1. Maps the occurrence of user-specified keywords or groups of keywords on the the sentences of a text; 2. Calculates the mean sentence length in words and the mean word length in syllables of the sentences in which each keyword or group of keywords occurs.
7.	LOG	1. Prints text with lines and sentences numbered in sequence; 2. Compiles a log indicating the line on which each sentence begins.

KOETHER & COKE
AERA, 1973

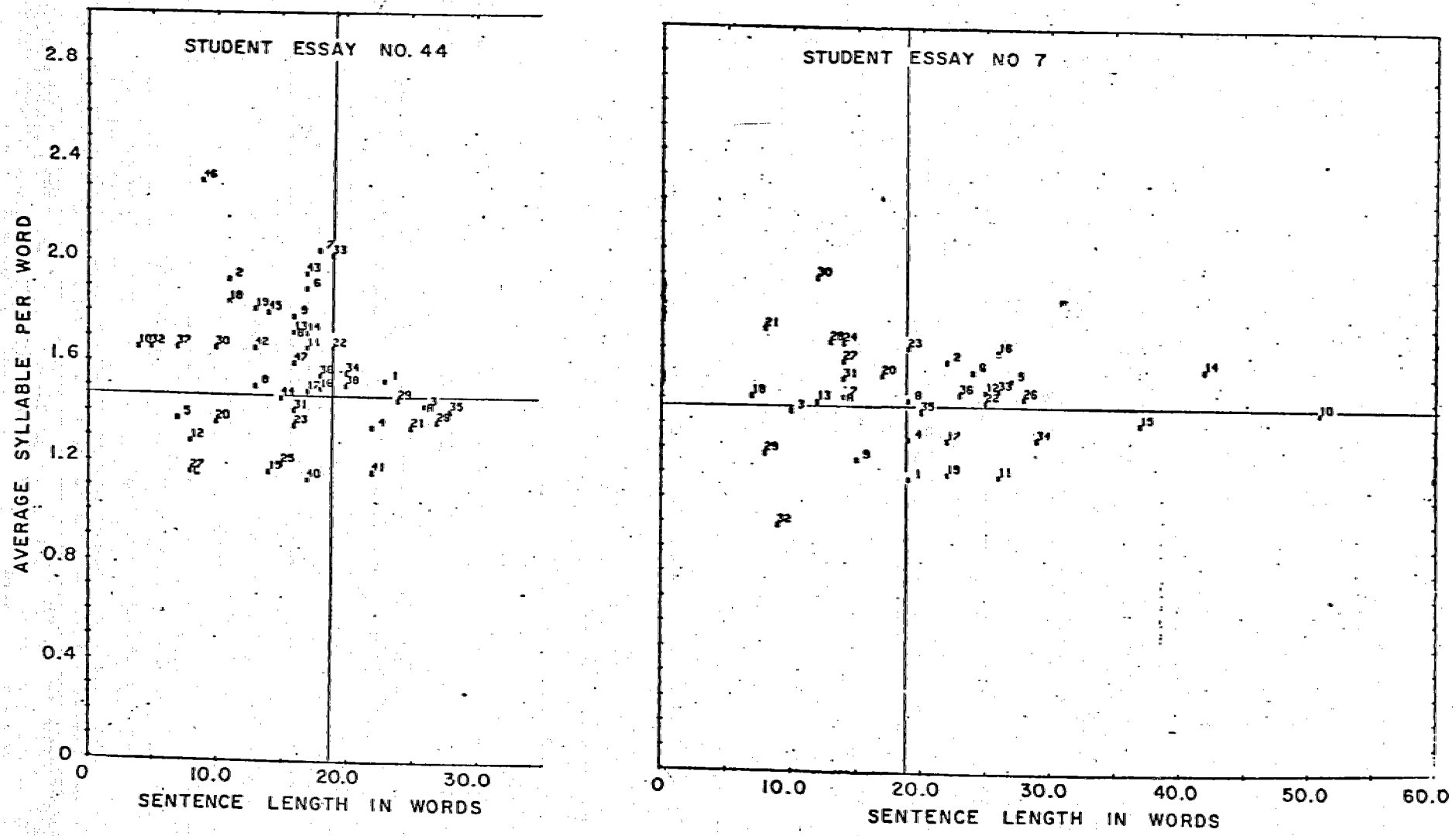


FIG. 5
THE JOINT DISTRIBUTION OF SENTENCE LENGTH AND THE AVERAGE LENGTH OF A WORD IN A SENTENCE FOR TWO STUDENT ESSAYS ABOUT LEATHER PROCESSING

The manager's advice to Tom was -

a. to continue working with Sears and not to go to college.

b. to graduate from high school and join the service.

→ c. to do well in school and attend a big state college.

d. to attend the local liberal arts college.

"This is the point," the manager concluded, "after you work here next summer, I think you better go to college. Not to some little liberal arts school like we have here in town, but a big state college where you can study business administration. Now you get an application this fall and pay attention to your grades this year. Next summer you save your money and I'll see about getting you a part-time job with Sears near the school."

FIG. 6

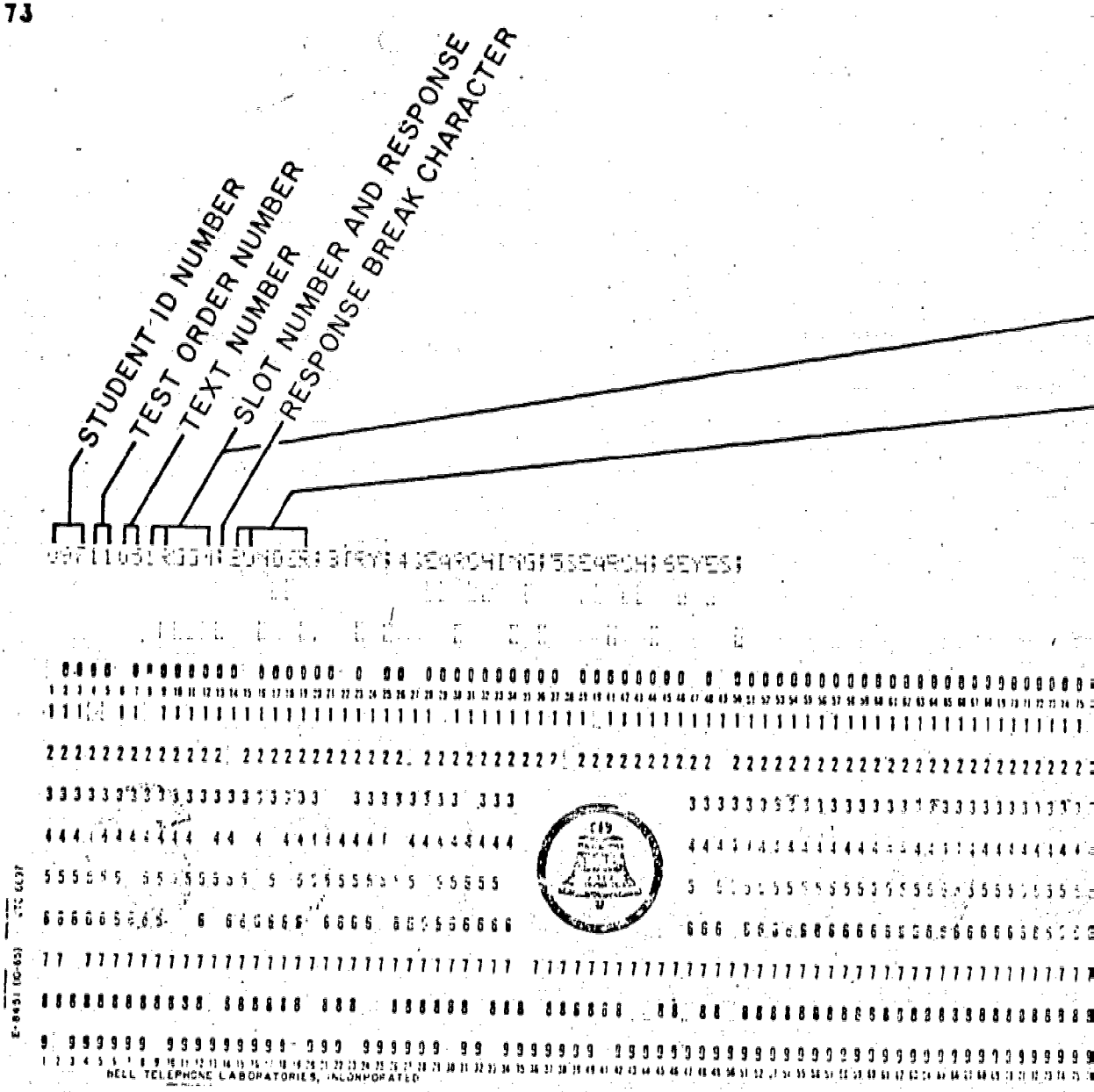
ILLUSTRATION OF SUBPHRASES OF ALTERNATIVES FOR ONE ITEM OF A MULTIPLE—CHOICE TEST. THE TEST ITEM IS ON THE LEFT; A PARAGRAPH FROM THE TEST'S READING SELECTION IS ON THE RIGHT. ALTERNATIVE "C" IS CORRECT. ARROWS INDICATE THE LOCATIONS OF THE SUBPHRASES IN THE TEXT.

TABLE 3

The ratio of the number of phrases found in a reading selection to the number of different phrases searched for in the selection as a function of the source of the phrases in the selection's multiple-choice test.

READING SELECTION	SOURCE OF THE PHRASES IN THE MULTIPLE-CHOICE TEST	
	ERROR ALTERNATIVES	CORRECT ALTERNATIVES
Pre-lesson	3.15	2.06
Unit 2	3.02	2.80
Unit 6	2.18	1.99
Post-lesson	1.98	2.09

KOETHER & COKE
AERA, 1973



RESPONSE CARD FOR RESPONSES TO A SINGLE TEXT GIVEN BY A SINGLE SUBJECT

FIG. 7
SAMPLE OF CLOZE-TEST SCORING

FREQUENCY DISTRIBUTION OF RESPONSES GIVEN FOR EACH DELETED WORD (SLOT) IN A TEXT

TEXT 5

SLOT: 1 BLANKS = 0 TYPES = 2

2.	ROAM	1
3.	ROOM	37

SLOT: 2 BLANKS = 0 TYPES = 7

2.	UNDER	28
3.	INSIDE	2
4.	UNDERNEATH	1
5.	IN	2
6.	UNDERNEALTH	1
7.	BENEATH	3
8.	BEHIND	1

SLOT: 3 BLANKS = 2 TYPES = 7

2.	BEGIN	8
3.	TRY	15
4.	START	7
5.	GO	2
6.	ROAM	1
7.	IN	2
8.	PROCEED	1

SLOT: 4 BLANKS = 1 TYPES = 4

2.	LOOKING	21
3.	HUNTING	9
4.	LOOK	1
5.	SEARCHING	6

SLOT: 5 BLANKS = 1 TYPES = 10

2.	SESSION	1
3.	TURN	4
4.	GAME	11
5.	PERSON	3
6.	TIME	8
7.	SEARCH	3
8.	TEML	1
9.	ROUND	4
10.	HUNT	1
11.	INDIVIDUAL	1

SLOT: 6 BLANKS = 2 TYPES = 6

2.	EYES	22
3.	HEAD	9
4.	LEAD	1
5.	BERICL	1
6.	FACE	1
7.	BACK	2