DOCUMENT RESUME

ED 038 023 95 EM 007 914

TITLE	Progress Report; Stanford Program in Computer-Assisted Instruction for the Period July 1, 1969 to September 30, 1969.
INSTITUTION	Stanford Univ., Calif. Inst. for Mathematical Studies in Social Science.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C.
PUB DATE	30 Sep 69
CONTRACT	OEC-0-8-001209-1806
GRANT	OEG-0-9-140401-4147
NOTE	47p.
EDRS PRICE	EDRS Price MF-$0.25 HC-$2.45
DESCRIPTORS	*Annual Reports, *Computer Assisted Instruction, Program Descriptions, Programing, Research and Development Centers

ABSTRACT

During this reporting period, the Drill-and-Practice Mathematics Program implemented a pilot version of the fixed strands curriculum. Data is presented illustrating the use of the program at the various participating schools. Adjustments were made to the Drill-and-Practice Reading Program Curriculum. Extensive recoding of the curriculum of the Logic and Algebra programs was undertaken. An analysis of various syntactical operations was made and work proceeded on de-bugging the program. Some revisions, changes, and additions were made to the Second-year Russian program. Work continued on three computer-assisted instruction (CAI) systems of teaching programing--AID, SIMPER, and LOGO. These and some of the other programs needed revising in order to be run on the PDP-10 computer. Some basic research into the field of CAI itself was carried on. Work continued on updating and perfecting the hardware and software systems at the Institute. The report also contains a brief outline of future plans and a list of lectures and publications by members of the Institute staff. (JY)

PROGRESS REPORT

STANFORD PROGRAM IN COMPUTER-ASSISTED INSTRUCTION

for the period

JULY 1, 1969 to SEPTEMBER 30, 1969

Office of Education Contract OEC-0-8-001209-1806

National Aeronautics and Space Administration
Grant NGR-05-020-244

National Science Foundation Grant NSFG-18709

National Science Foundation Grant NSF GJ-197

National Science Foundation Grant NSFG J-443X

Stanford Subcontract under Office of Education
Grant No. OEG-9-8-685083-0041
with Ravenswood City School District

Stanford Contract under National Science Foundation
Grant NSF GY-5308
with Tennessee A. and I. State University

Office of Education Grant OEG-0-9-140401-4147

Gallaudet College Contract

INSTITUTE FOR MATHEMATICAL STUDIES IN THE SOCIAL SCIENCES
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

Table of Contents

I. Major Activities of the Reporting Period

A. Drill-and-practice Mathematics Program

1. Strand Program

A pilot version of the strand program, referred to as fixed strands, has been written and is being implemented during the academic year 1969-70. The pilot version differs from the full version in two major respects; the problems are written and stored in the computer rather than generated on-line, and the rule for sampling problems has been simplified, so that only the objective distribution determines the proportion of problems from each strand presented to the student. In addition, a special function for movement through the strands has been defined for the fixed strands.

Implementation of the fixed strands has two important advantages. First the massive task of writing the computer program to run the strands has been separated into two parts, and the basic program can be debugged and put into operation before adding the subroutines necessary to generate problems on-line. Second, data can be obtained that bear directly on many of the assumptions which had to be made to write the strand program. These include assumptions about latencies, error rates, and homogeneity of equivalence classes.

Fixed Strand Curriculum

The fixed strand curriculum retains the basic structure of the strand curriculum, i.e., the partition of the substance of elementary-level mathematics into 15 strands, and each strand into a set of equivalence classes. For the fixed strand curriculum, each strand consists of a set of problems in a fixed order, with the order within each equivalence class determined by randomizing the order of a set of problems written to satisfy the definition of the class.

Number of problems in a class. The number of problems written for each equivalence class depends on the expected number of problems to be worked in each strand, at each half-grade level, and the error rate assumed to describe performance. Thus, if the average latency for a problem, L, the total time spent at the computer, T, and the percentage of problems for a given strand, P, are known for a given half year, the expected number of problems, EP, is given by

the formula

$$EP = \frac{T}{L} \times P \quad (EP = \frac{T}{L} \times P) \tag{1}$$

and the number of problems written, PW, is given by the formula

$$PW = \frac{EP}{1+2q} \tag{2}$$

where q is the assumed error rate. The second formula requires some comment. In the basic strand program, when a student makes an error he is immediately presented with an additional problem from the same equivalence class, and this problem is not counted, for purposes of meeting the criterion performance necessary to advance to the next class. This fact (the presentation of the extra problem) was incorporated into formula (2), although in the context of the fixed strands, where problem order is fixed, this procedure has a quite different meaning and will not be followed. An error rate, q, of .4 was chosen on an intuitive basis. It is higher than the average error rate found for the drill-and-practice program. However, it was felt that since all problems would be presented in a mixed format, the error rate would quite likely be higher than that previously found.

The number of problems written for each strand for each half year is presented in Table 1.

Rule for sampling problems. The objective probability distribution, $f_p(j)$, determines in each instance the strand from which a student receives a problem. The actual problem the student receives depends on his performance on the past several problems from that strand in the following way: the direction of movement on the strand depends on whether the last problem worked was correct or incorrect, and the distance moved depends on the length of the string of correct or incorrect responses. A string of responses is said to start at a change in response, so that if problem k is the first in a string, and the response is correct (C), then the response on problem k-1 was incorrect (I). The unit of distance is one problem. The details of the movement scheme are shown in Table 2. The set of distances chosen is a subset of the Fibonacci sequence, $F_{n+2} = F_n + F_{n+1}$, which starts with the third term of the sequence, and ends with the sixth term.

TABLE 1

Number of Problems Written for Each Strand for Each Half Year

Strand	Half-year											
	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	6.5
Number Concepts	250	120	120	95	85	35	55	50	55	70	100	60
Horizontal Addition	220	190	190	80	100	65						
Horizontal Subtraction	125	95	115	80	30							
Vertical Addition	95	80	85	170	150	50	70	15	20	10		
Vertical Subtraction		80	85	140	150	65	70	15	20	10		
Equations		70	70	100	120	100	140	50	45	60	60	65
Measurement		40	60	50	40	50	50	50	40	40	40	40
Horizontal Multiplication				80	45	120	70	40	45			
Laws of Arithmetic					30	50	40	30	30	30	10	15
Vertical Multiplication						120	40	100	45	20	20	15
Division						150	70	100	90	30	40	15
Fractions						35	25	160	120	160	150	160
Decimals							55	40	30	70	50	240
Negative Numbers											20	25

TABLE 2

Movement Scheme Adopted for Fixed Strands

Type of Response	Length of String	Distance Moved
Correct	1	+1
	2	+2
	3	+3
	4	+5
	>4	+5
Incorrect	1	-1
	2	-2
	3	-3
	4	-4
	>4	-5

Movement on the strand structure. The movement scheme adopted (shown in Table 2) has an important feature: it is not necessary to retain a history of a student's performance prior to the last worked problem to determine future movement. All that need be kept is a record of a signed number representing the distance moved. Thus, if the number kept for a student is -3, and he works problem k correctly, the number kept for him will be +1, and he will next be presented with problem k+1. Or, if the number retained is +3, and he works problem k correctly, the number kept will be +5, and the next problem presented will be k+5.

One of the objectives of the strand program is to examine in detail individual differences in performance. For this purpose it is important to have a movement scheme which allows great variation in movement rates. The scheme chosen has this property. A student giving a correct response to every problem would be able to move through an entire year's work in about six weeks.

Initial grade placement on strands. The grade placement obtained for each student from his performance on the computation section of the SAT, or actual grade in school, whichever is lower, will be used to determine initial placement on the strands. However, since we have no evidence that this will be an adequate criterion for assigning grade placement, a scheme will be used to accomplish large changes in grade placement, during the first 20 days a student is signed onto the system. If, in any single session, a student's error rate is greater than .8, he will immediately be moved down half a grade level. If the student's error rate is less than .05, he will be moved up half a grade level.[1] For students whose error rates fall between .8 and .05, movement will be governed entirely by the scheme already described.

2. Use of the System in Schools

The total number of drill-and-practice lessons in mathematics during the summer months was 14,485. Most of the lessons were taken during the month of July and the first day of August. During the approximate period of August 2 to September 1, no schools were in session and no students took lessons on the system. In early September, schools reopened and classes were gradually restarted as indicated in the following tables.

---

[1] Specifically, the student will be moved to the beginning of a half grade which is greater than one half a grade, but less than a full grade level from his current position, so that the new position will always be at the first equivalence class of a half grade.

Tables 3 and 4 give the total number of lessons taken each day in each area and the total lessons for each area each month. August 1 is included as the last entry in Table 3 along with the data for July. The number of reading and logic lessons taken is also shown in Table 3 and may be referenced in later sections of this report.

Tables 5 and 6 give a daily class breakdown of the number of lessons taken. The numbers under the headings "Calif. (10,0)" indicate class numbers. The class numbers for Brentwood schools were 6, 7, 8, 9, and 11, etc.

3. California Schools

The entry "California Schools" during this time period includes special accounts, hourly users, and demonstrations. Although Brentwood is in California, it is recorded separately.

The schools in the Ravenswood School District, East Palo Alto, California were Belle Haven, Brentwood, Kavanaugh, and Willow. The number of lessons taken by each school is shown in Tables 5 and 6. Table 7 lists the concept blocks selected by teachers for each class, the class number, the number of students in each class, and the number of students working in each block on August 1 or the end of their summer session. Several interesting features of the program are documented here. One is the variety of the block sequences selected for different classes. Second is the class sizes for which different sequences were chosen, and third is the students' progress through the lessons as indicated by their position at the end of the term. For example, of the 30 students in class 52 in Belle Haven School, the concept blocks selected were 102, 106, 110, 111, 112, 113, 114. Each block contains work for seven sessions on school days if one lesson is taken each day. One student was working in Block 112 which means he completed at least 28 lessons on the system by the end of the term. The same entry also shows that five students did not complete the first block. These students either enrolled and dropped out of summer school all together or for some other reason failed to take daily lessons as intended. In the future, more care will be taken to include only those students who complete a reasonable amount of the program in comparative studies.

TABLE 3

DAILY SUCCESSFUL RUNS ON DRILLS
JULY, 1969

DAY	RAVNS.	CALIF.	TENN.	SEATTLE	TOTAL	READ.	LOGIC
7-1	406	28	41	61	536	99	0
7-2	609	39	15	54	717	134	0
7-3	493	22	43	41	599	89	8
7-7	626	8	3	42	679	176	22
7-8	833	5	2	43	883	312	23
7-9	812	2	48	48	910	332	9
7-10	921	16	70	55	1062	378	11
7-11	824	13	0	56	893	309	14
7-14	826	0	89	50	965	300	16
7-15	797	18	52	39	906	282	31
7-16	666	17	55	41	779	195	41
7-17	786	13	23	37	859	277	51
7-18	651	11	69	36	767	188	65
7-22	739	13	120	42	914	226	71
7-23	851	18	81	47	997	270	9
7-24	892	13	98	47	1050	332	19
7-25	853	22	0	45	920	301	20
7-28	821	6	0	18	845	352	10
7-29	674	19	0	0	693	217	4
7-30	648	4	93	10	755	213	6
7-31	685	9	23	20	737	228	15
8-1	441	33	0	20	494	132	9
	15854	329	925	852	17960	5342	454

TABLE 4

DAILY SUCCESSFUL RUNS ON DRILLS
SEPTEMBER 1969

DAY	CALIF.	RAVN.	WASH.	TENN.	MED.CTR.	TOTAL
9-2	1	1				2
9-3	0	2				2
9-4	0	12	3			15
9-5	1	15	0			16
9-8	5	28	0			33
9-9	5	*	1			6
9-10	4	47	4			55
9-11	22	5	12			39
9-12	1	110	16	57		184
9-15	29	8	91	0		128
9-16	2	14	20	2		38
9-17	0	85	106	0		191
9-18	0	205	32	0		237
9-19	0	242	67	0		309
9-22	0	177	135	304		616
9-23	5	130	27	272		434
9-24	2	290	163	316		771
9-25	17	201	43	150		411
9-26	6	381	207	221		815
9-29	1	550	158	390		1099
9-30	2	648	64	407	3	1124
	103	3151	1149	2119	3	6525

TABLE 5

MONTHLY DISTRIBUTION OF DRILLS RUN PER SCHOOL
JULY, 1969

	1	2	3	7	8	9	10	11	14	15	16	17	18	22	23	24	25	28	29	30	31	1	TOTAL
CALIF.																							
10	0	9	9	1	0	0	0	0	0	5	0	0	0	3	0	0	13	0	9	1	3	2	55
0	28	30	13	7	5	2	16	13	0	13	17	13	11	10	12	13	9	6	10	3	6	31	274
BRENTWOOD																							
6	95	547	153	136	146	165	163	131	181	148	152	143	129	162	171	156	156	134	136	150	119	91	3564
7	155	0	186	256	368	371	405	394	372	371	303	390	273	288	363	363	347	366	225	218	222	143	6369
8	85	0	0	100	162	144	207	172	184	182	140	185	159	184	230	239	276	240	216	190	249	103	3646
9	7	0	82	71	87	86	97	96	85	80	66	70	58	89	80	96	73	73	92	80	74	90	1622
11	64	62	72	63	70	46	49	31	4	16	5	9	32	16	7	48	1	8	5	10	22	14	653
TENNESSEE																							
3	14	15	16	0	1	44	59	0	40	51	35	4	23	69	11	40	0	0	0	0	0	0	422
12	27	0	27	3	1	4	11	0	49	1	20	19	46	51	70	58	0	0	0	93	23	0	503
SEATTLE																							
1	40	34	24	25	20	24	31	33	28	28	22	19	12	26	28	24	30	0	0	0	0	0	448
2	21	20	17	17	20	22	24	23	21	11	19	16	21	16	18	19	15	18	0	10	20	20	388
13					3	2	0	0	1	0	0	2	3	0	1	4	0	0	0	0	0	0	16
	536	717	599	679	883	910	1062	893	965	906	779	859	767	914	997	1050	920	845	693	648	737	494	17960

LOGIC AND READING COUNT
JULY, 1969

LOGIC RUNS

DATE	3	7	8	9	10	11	14	15	16	17	18	22	23	24	25	28	29	30	31	1	TOTAL
	8	22	23	8	11	14	16	31	41	51	65	71	9	19	20	10	4	6	15	9	454

READING

DATE	1	2	3	7	8	9	10	11	14	15	16	17	18	22	23	24	25	28	29	30	31	1	TOTAL
0	27	134	26	44	58	34	49	32	3	16	4	6	32	7	10	31	4	2	4	7	12	21	563
6	14	0	19	9	25	26	48	17	47	28	28	27	21	34	45	56	35	47	27	49	41	30	674
7	39	0	25	105	177	217	198	200	191	187	139	188	116	143	150	169	168	211	104	96	98	57	2979
8	19	0	19	9	38	33	57	38	40	36	19	44	15	23	50	57	75	82	58	54	75	23	866
9	0	0	0	9	14	22	24	21	19	15	5	12	4	19	15	19	19	10	23	7	2	1	260
	99	134	89	176	312	332	378	308	300	282	195	277	188	226	270	332	301	352	217	213	228	132	5342

TABLE 6

MONTHLY DISTRIBUTION OF DRILLS RUN PER SCHOOL
SEPTEMBER 1969

	2	3	4	5	8	9	10	11	12	15	16	17	18	19	22	23	24	25	26	29	30	TOTAL
DEMOS & SPECIAL ACCTS.																						
0	1	0	0	1	5	5	3	3	0	26	0	0	0	0	0	1	1	16	6	1	0	69
1	0	0	0	0	0	0	1	19	1	3	2	0	0	0	0	4	1	1	0	0	2	34
CALIF. SCHOOLS																						
6																					3	3
BRENTWOOD																						
2	0	0	0	1	0	*	0	0	0	0	0	43	182	176	161	98	266	151	302	411	487	2278
3	0	0	0	0	0	*	0	0	0	0	0	0	0	1	0	6	12	9	48	86	99	261
4	0	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	1	2	12	14	28	*	47	5	110	8	14	42	23	65	16	26	12	41	31	53	62	612
WASHINGTON																						
20	0	0	0	0	0	0	0	10	12	90	20	104	27	64	135	27	163	43	207	156	64	1122
29	0	0	3	0	0	1	4	2	4	1	0	2	5	3	0	0	0	0	0	2	0	27
TENNESSEE																						
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	42	81	168	264	319	896
39	0	0	0	0	0	0	0	0	57	0	2	0	0	0	304	250	274	69	53	126	88	1223
	2	2	15	16	33	6	55	39	184	128	38	191	237	309	616	454	771	411	815	1099	1124	6525

* SCHOOL HOLIDAY

LOGIC AND READING COUNT
SEPTEMBER 1969

LOGIC RUNS DATE	2	3	4	5	8	9	10	11	12	15	16	17	18	19	22	23	24	25	26	29	30	TOTAL
	0	0	0	0	4	5	5	0	0	0	0	0	0	0	0	14	12	23	23	0	18	110

SLAKER DATE	2	3	4	5	8	9	10	11	12	15	16	17	18	19	22	23	24	25	26	29	30	TOTAL FRAMES
	0	0	418	1004	1441	*	3430	2883	2794	6669	5629	3625	1691	5348	5295	3924	2960	5621	3689	3713	5553	65,707

* SCHOOL HOLIDAY

TABLE I

CLASS CONCEPT POSITION AT THE END OF SUMMER SCHOOL - 1969

SEATTLE CLASSES

CONCEPT ORDER	CLASS NO.	NO. OF STUDENTS	STUDENTS PER CONCEPT
NO SET ORDER	14	9	101 113 210 214 1 3 2 3
NO SET ORDER	15	8	301 307 313 310 1 2 3 1
NO SET ORDER	19	6	101 102 103 104 2 1 1 1
NO SET ORDER	20	2	114 118 1 1
NO SET ORDER	21	5	601 621 4 1
NO SET ORDER	22	3	101 102 104 1 1 1
NO SET ORDER	23	3	401 421 1 1
NO SET ORDER	24	3	601 604 1 1
NO SET ORDER	25	4	101 105 108 2 1 1
NO SET ORDER	26	2	501 505 1 1
NO SET ORDER	27	2	101 107 1 1
NO SET ORDER	28	3	201 222 1 2
NO SET ORDER	29	4	407 410 411 413 1 1 1 1
NO SET ORDER	30	6	101 104 106 115 123 2 1 1 1 1
NO SET ORDER	31	2	201 214 1 1
NO SET ORDER	32	1	204
NO SET ORDER	33	1	504 1

TENNESSEE CLASSES

CONCEPT ORDER	CLASS NO.	NO. OF STUDENTS	STUDENTS PER CONCEPT
401 - 410	40	14	401 402 403 404 2 2 8 2
501 - 510	41	12	501 502 506 507 508 4 1 2 2 3
701 - 711	43	4	702 703 3 1

BELLE HAVEN CLASSES

CONCEPT ORDER	CLASS NO.	NO. OF STUDENTS	STUDENTS PER CONCEPT
102,106,110-114	52	30	102 106 110 111 112 5 5 9 4 1
102,106,110-114	53	25	102 106 110 111 10 4 7 4
201,202,206-208,211 212	54	16	201 202 206 207 9 3 3 1
201,202,206-208,211 212	55	16	201 202 206 2 3 3
525,528,403,314,320 306,309	56	17	525 528 403 314 6 3 6 2
525,528,403,314,320 306,309	57	17	525 528 403 1 4 6
525,528,502,410,414, 404,406	58	11	525 528 502 2 3 2
525,528,502,410,414 404,406	59	23	525 528 502 410 7 3 11 2
525,528,502,524,507 503,508	60	12	528 502 524 1 6 2
525,528,502,524,507 503,508	61	10	525 528 502 524 2 2 4 2
525,528,502,624,604 601,609	62	26	525 528 502 624 9 7 8 2
525,528,502,624,604 601,609	63	6	525 502 624 3 1 2
101-106	64	10	101 102 103 104 2 1 6 1

BRENTWOOD CLASSES

CONCEPT ORDER	CLASS NO.	NO. OF STUDENTS	STUDENTS PER CONCEPT
102	69	4	102 4
102,101	70	39	102 101 37 1
102,106,110-114	71	31	102 106 110 6 9 13
102,106,110-114	72	29	102 106 110 111 112 2 4 12 1 1
201,202,206-208,211 212	73	34	201 202 206 207 6 5 14 9
201,202,206-208,211 212	74	40	201 202 206 207 208 8 11 6 10 5
525,528,403,314,320 306,309	75	8	525 403 314 3 2 1
525,528,403,314,320 306,309	76	32	525 528 403 314 11 1 7 6
525,528,502,410,414 404,406	77	35	525 528 502 410 414 7 5 11 7 2
502,503,507,508,528 506	78	37	502 503 507 23 13 1
525,528,502,624,604 601,609	79	31	525 528 502 624 604 16 4 7 3 1
525,528,502,624,604 601 609	80	9	525 528 502 2 2 5
525,528,502,410,414 404,406	82	7	525 528 502 410 2 2 2 1

KAWAMAUGH CLASSES

CONCEPT ORDER	CLASS NO.	NO. OF STUDENTS	STUDENTS PER CONCEPT
102,101	85	34	102 101 33 1
102,101	86	38	102 101 36 1
102,106,110,111,112 113,114	87	38	102 106 110 111 112 6 11 13 4 2
102,106,110,111,112 113,114	88	12	102 106 110 114 5 1 2 1
201,202,206,207,208 211,212	89	28	201 202 206 207 208 14 1 10 1 1
201,202,206,207,208 211,212	90	36	201 202 206 207 208 15 6 11 3 1
525,528,403,314,320 306,309	91	36	525 528 403 314 309 19 5 10 1 1
525,528,403,314,320 306,309	92	15	525 403 314 306 3 6 2 1
525,528,502,410,414 404,406	93	16	525 528 502 410 5 1 6 1
525,528,502,410,414 404,406	94	11	525 528 502 414 2 5 3 1
525,528,502,524,507 503,508	95	23	525 528 502 524 507 7 2 10 3 1
525,528,502,524,507, 503,508	96	11	525 528 502 524 507 2 2 2 2 1
525,528,502,624,604, 601,609	97	27	525 528 502 624 604 609 10 4 9 2 1 2

WILLOW CLASSES

CONCEPT ORDER	CLASS NO.	NO. OF STUDENTS	STUDENTS PER CONCEPT
102,106,110,111,112 113,114	101	34	102 106 110 111 22 5 3 3
201,202,206,207,208 211,212	102	24	201 202 206 207 4 6 9 5
502,524	103	29	502 524 000 15 9 1
403,314	104	38	403 314 31 6
403,314,320,306,309	105	7	403 314 5 2

4. Seattle Classes

Two teletype terminals were used in Seattle, one in the Experimental Education Unit of the University of Washington and the other in a nearby school for deaf and handicapped children. The entry in Table 7 "no set order" indicates that a considerable reordering of lesson blocks took place. In several cases the students took the lessons a second time. The variety of levels of lesson difficulty is shown in the position of the students at the summer session. Block numbers whose first digit is 1 are at first-grade level. Block numbers beginning with the digit 2 are second-grade level, etc. Also notice the small number of students in each class. This indicates that the lessons were prescribed for individuals or very small groups and is an indication of the degree of individualization possible with the flexibility offered by a CAI system. The researchers in Seattle made almost daily adjustments in lesson sequences.

5. Tennessee Classes

The number of students at Tennessee A. and I. State University enrolled in CAI classes was small, 30 in all. These were assigned in three different "computer" classes as shown in Table 7. Each class was at a different level of achievement as indicated by the first digits of each concept block number. There was considerable variety in the number of lessons completed by different students.

6. Washington, D. C. Schools

Kendall School for the Deaf in Washington, D. C. did not have a summer session. Regular classes began in early September as shown in Table 6. To date their achievement has been very satisfactory, and they have expressed considerable interest in programming courses in several content areas other than mathematics.

During the 1968-69 school year, preliminary results reported by Kendall School on achievement of both high school and elementary-school students were very encouraging. Approximately 60 per cent of the project population were of high school age. The remaining 40 per cent consisted of students from several age levels, some as young as 9 years old. According to the information we have been given, 30 per cent of the 90 students who participated in this project may be classified as multiply-handicapped deaf children from the inner city. All students in the secondary and elementary grades took daily lessons in arithmetic, and selected students in junior and senior high school took the tutorial logic and algebra program.

The data from the first five months of operation in elementary mathematics at Kendall School are summarized by grade equivalence scores in Figure 1. Note that the normal expectancy for grade-placement gain for deaf children is one half the number of calendar months. That is, deaf children gain, on the average, one-half year on standardized achievement tests for each year in school. As shown in Figure 1, 64 per cent of the students achieved a gain of nearly a half year in just five school months. The tests on which these data are based are the Stanford Achievement Test, the Metropolitan Achievement Test, and the Wide Range Achievement Test. The data graphed in Figure 1 are the mean grade equivalence scores on the computational skills sections from the three tests.

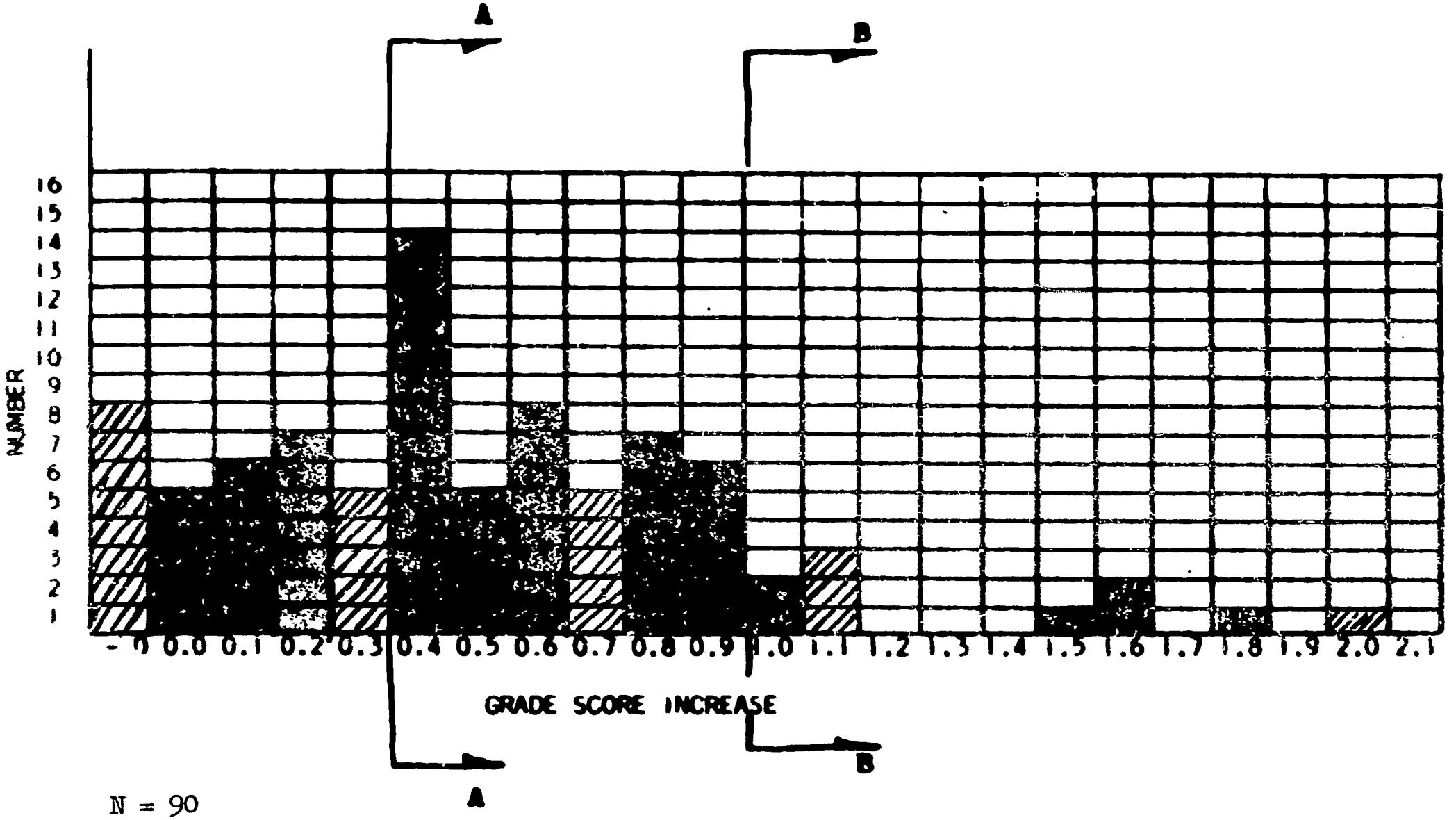
The data indicate that the increase in performance on the mathematics section of the battery was greater than the increase in performance on the entire battery.

Pupil reaction to the program as reported by the Kendall School staff has been highly favorable. Students were eager to put forth as much effort as they could in attempting to improve their performances. The looks of disappointment on the faces of students who did not get to use the CAI on a particular day, for whatever reason, were highly indicative of the generally positive reactions to the project.

Speed, concentration, and efficiency generally seemed to improve. For example, many students apparently stopped counting on their fingers and were able to do more problems "in their heads." They also were able to learn problems with which they had had difficulty by using CAI. Many demanded to know how to correct their errors, which in itself was unusual. Further, these students demonstrated amazing "stick-to-it-iveness."

Children were observed who continued working problems even though making all incorrect responses. These were often children who would have given up on similar problems in the classroom or on other programmed instructional programs.

Students also demonstrated some self-discovery as a result of meeting difficulties on a given lesson. One boy did not learn the concept of a particular day's lesson in class; he got the first 14 problems on the CAI lesson wrong but the last 2 problems right and came away from the machine able to demonstrate his understanding of the new concept.



N = 90

A. 64% ACHIEVED MORE THAN OR EQUAL TO A 0.4 GRADE LEVEL GAIN IN 5 SCHOOL MONTHS.

B. 10.5% ACHIEVED MORE THAN OR EQUAL TO A 1.0 GRADE LEVEL GAIN IN 5 SCHOOL MONTHS

Fig. 1. Average grade placement gain scores on three standardized achievment tests for students at Kendall School.

Students tended to personalize the teletypes. It was not unusual to see them apologize to a machine for hitting a wrong key. On the other hand, of course, they could be rude to it without fear of punishment. This freedom of expression occasionally had its faults. One boy became so angry that he punched and damaged a machine beyond repair at Kendall School.

Within two days all pupils grasped how to operate the teletypes. The simple language used in the lessons presented no problems. Limited trials with kindergarten children showed they can learn to use the equipment for drill as well, although they appeared to need more supervision.

Many pupils seemed to develop a more mature attitude toward learning. Pupils who at first became angry and quit when they were wrong began to persist through the problem. After looking over their pretest with a teacher or supervisor, many students went home and, of their own volition, practiced for the next day.

Other important indications of favorable student reaction include the generally more positive response a student demonstrated when the computer typed "NO, TRY AGAIN," than when a teacher told him he was wrong. Another was the immediate response of the computer to student errors which is not possible for a teacher in a group drill-and-practice session in a classroom.

After the first five months' operation, the staff reported that "All of these reactions, positive and negative, can be traced to the extremely high motivation of the students. The novelty did not wear off. They were extremely enthusiastic and usually proud of themselves."

B. Drill-and-practice Reading Program

1. Curriculum, Additions, and Adjustments

The reading staff took advantage of the time following the summer school run and the end of this reporting period to make adjustments in the curriculum. Originally the curriculum adhered strictly to vocabulary in the basal texts used in the Ravenswood District's primary reading program, and as such, was programmed as three separate series. However, based on observation, quality adjustments in the audio and systems, and student reaction, the staff chose to delete certain sections in particular strands to avoid repetition of words within strands. Using existing utility programs, word lists were assembled

from major basic reading texts as well as from three recognized sight-word lists. These vocabulary lists were compared and screened for common occurrence and order of introduction. This new listing provided the basis for the vocabulary strand and resulting phonics and spelling patterns in the reading program. In its revised form, the Stanford reading program consists of a curriculum that can now complement any classroom reading series.

The inputting and preprocessing of the revised curriculum in the sight-word, phonics, and spelling strands, and the two comprehension strands, detailed in this report, were completed.

The strands of the reading curriculum then, as proposed in earlier reports, were completed and ready for operational use in the 1969-70 school year. An additional strand, language arts, is outlined as a sixth strand for the program, and is planned to correlate closely with classroom activities.

2. Systems

During this quarter, all software for the reading program was rewritten for greater efficiency and compatibility with the DEC Series 4 monitor, the executive program that provides for time-sharing and re-entry among user programs. The preprocessing, audio recording, and software needed for the two new comprehension strands were completed. The reading programming staff also began writing the new student record keeping and reporting routines to be used in common by all IMSSS curriculums. Formats of the reading class and individual student reports have not been changed, but the appropriate programs are to be compatible with the re-entrant capability of the Series 4 monitor.

3. Word Meaning Strand

The newly prepared material was added to the curriculum in the form of two strands--a comprehension categories strand and a comprehension sentence strand. These strands provide practice on the meaning of those words introduced in the classroom and mastered by the student in the sight-word strand. It is a means, therefore, to maintain that vocabulary.

In the comprehension category strand, one of several categories is associated with each of the words in the section. A presentation consists of the display of three words followed by a request to type the word that is in a given category as in the following example:

Display	Audio
HOUSE CAT GREEN	Type the word that stands for an animal.

The order of the three words in the display is random and the target word, with its associated category, is chosen at random from the displayed words. If the program selects "green" as the target word for presentation, the audio requests, "Type the word that stands for a color." "House" generates a request to identify the word "that is a place."

In the comprehension sentence strand, a section consists of three sentences each with one word missing. Associated with each sentence is a word that fits into the empty "slot" correctly and two distractors. One of the distractors is of the correct form class, but is either semantically unacceptable or syntactically unacceptable in that it breaks a subcategorization rule in the sense of Chomsky,[2] and the second distractor is unacceptable both semantically and syntactically. The format for this strand is the following:

Display	Audio
MAD DRIVE SWIM TIM WILL --- THE CAR	Type the word that goes in the sentence.

This exercise provides use of meaning clues or inference from sentence context and permits greater variety within the learning task.

4. Support Program

To facilitate preprocessing of the reading curriculum, numerous details and programs were incorporated into a procedure that yielded (from the original curriculum file) a group of audio directories, a curriculum dictionary, and the necessary components for the preprocessing itself. Briefly, this procedure can be outlined as follows:

1. Modify (if needed) and proofread the curriculum file, taping a backup version.
2. Convert the current audio directory into a machine-readable (binary) format. ("Textmaker" program)
3. Create a list of curriculum items with audio numbers using the output of 1 and 2 as input. ("Getter" program)

---

[2]Chomsky, N. Aspects of the theory of syntax. Cambridge, Mass.: M.I.T. Press, 1965.

4. Assign numbers to items with no audio, record those items and update both the output of 3 and the master audio directory.
5. Sort the curriculum-ordered audio directory (the output of 3) into alphabetical order, removing duplicate items (used in various parts of the curriculum) from the output. ("Alfsort" program)
6. Sort the output of 5 into numerical order. ("Numsort" program)
7. Create the curriculum dictionary:
   a. List all items without audio numbers by strands. ("Getter" program)
   b. Convert each strand list into machine-readable format. ("Text-maker" program)
   c. Mark the alphabetical curriculum audio direction with identifiers indicating in which strand(s) each item occurs, using the output (successively) of b. ("Marker" program)
   d. (optional) Prepare standard word lists for initial reading (e.g., Gates, Chall) in the same manner as curriculum strands and mark dictionary items with identifiers.
8. Review the curriculum dictionary to determine whether any words are overused.
9. Make final modifications in original curriculum input file, repeating steps 2 through 8 if a great many corrections are to be made (otherwise, simply update the affected files).
10. Make final listings and backup tapings of the following:
    original curriculum input
    curriculum dictionary
    master audio directory -- alphabetical
    master audio directory -- numerical
    curriculum-ordered audio directory
    curriculum audio directory -- numerical.

In connection with 7d above, a number of standard word lists were input and compared for the program. Besides the Lippincott, Ginn and BRL series, Allyn-Bacon, Scott, Foresman, Heath, and Singer series were entered. These lists are now available for comparison with curriculum items:

Dale List of 76 Words

Dale-Chall List of 3000 Familiar Words

Dolch List

Gates List of 187 Words.

All programs and files described above operate on the PDP-1. For final preprocessing, however, it became necessary to use the PDP-10. A major effort, therefore, was needed to adapt the PDP-1 preprocessor to the larger machine. This program takes as input the original curriculum file and master audio directory (transferred to the PDP-10) and outputs a PDP-10 file for each strand, ready for use by the reading driver.

5. Audio

Removal of one 2314 disk reduced to half the available storage space for audio. At the sampling rate of 72k bits per second, the number of messages fell below 2,000. By performing the support procedure previously mentioned, the user may decide which "sounds" to retain and reassign numbers and file sounds under new numbers.

Technology has since improved so that the bit rate has been cut in half, to 36k, increasing the capacity to over 3,000 messages. At the same time, efforts to improve the quality have proven successful.

C. Logic and Algebra Program

Extensive recoding of the curriculum was undertaken. The sentential connectives were replaced by their English counterparts. Thus, for example, '^' was replaced throughout by "and." Coding changes required so that the program could be run on the PDP-10 were also made.

A proposal to systematize error messages was considered with efforts directed primarily towards analyzing the student input in greater detail. Once such an analysis is carried out it will be possible both to gain insight into which concepts are difficult for the student and to send more pertinent error messages.

A quite thorough analysis is possible since the various syntactical operations of concern to the student are recursive. In particular, membership in the set of axioms, whether or not an expression is a term or formula of the symbolic language, and membership in the inference schemata are effectively calculable.

In the process of constructing a derivation or proof, a student's input, indicating his desire to either embark on a particular derivation procedure, take an instance of an axiom or previously proved theorem, or apply a rule of inference, is a three tuple $\langle\alpha,\beta,\gamma\rangle$, where α and γ are possibly the empty sentence and β is some code name for a derivation procedure, axiom, rule of inference, or variable for which a substitution is to be made. If the student were to type 3CA2, for example, he would be indicating his desire to transform line 3 of the derivation by means of the commute addition axiom: the terms around the second occurrence of +, counting from the left, are to be commuted in line 3.

Given such an instruction the computer responds by evaluating line 3 of the student's derivation. If there are at least two occurrences of + in line 3, the result of the operation is typed out. If the condition mentioned is not satisfied, "'+' does not occur twice." is typed out. A detailed description of how the student's inputs are to be analyzed and responded to has been written.

Theories with standard formalization, i.e., theories which can be formalized within the first-order predicate calculus, have, in recent years, been the subject of a thorough study. The mathematical and philosophical literature is filled with reports of studies of this most interesting set of theories. It is natural, then, to wish to extend the logic program to include a study of the first-order predicate calculus with identity and definite descriptions.

Our efforts to build on the firm basis of sentential logic, already developed and in operation, are described briefly. The symbolic language must, of course, be supplemented by the addition of notation for n-place predicate and operation symbols as well as notation for the quantifiers and definite descriptions. The formation rules, which determine what is to be counted as a well-formed formula or sentence, are to be extended in an appropriate fashion and additional rules of inference for the quantifiers are to be added.

Semantical notions such as logical consequence and logical truth do not provide any particular difficulty given the elegant mathematical treatment by Tarski and his students. The formal treatment of these notions, however, demands that we exploit all means available within a CAI framework.

Since predicate calculus has a valuable application in providing a logical analysis of arguments both in informal and technical discourse, it is expected that students acquire techniques for determining the validity of these arguments under their translation into the symbolic language.

Prior to the formal analysis of arguments, students will be afforded the opportunity to symbolize English arguments. For example,

Symbolize the sentence: If all men are wise and Socrates is a man, then Socrates is wise.

	(1)	M(x)
	(2)	W(x)
	(3)	M(s)
	(4)	W(s)
→1.2	(5)	(M(x) → W(x))
∧x5	(6)	∧x (M(x) → W(x))
∧6.3	(7)	(∧x (M(x) → W(x)) & M(s))
→7.4	(8)	((∧x (M(x) → W(x)) & M(s)) → W(s))

The idea behind this sort of exercise is quite simple. The appropriate atomic formulas of symbolic language are provided. The student gives instructions to the computer in Polish notation, i.e., parenthesis-free notation, and the computer types out the result obtained in the notation essentially of Principia Mathematica.

Since, by a thorem of the American logician, Alonzo Church, first-order theories are undecidable, a problem arises. There is no automatic method for determining when a symbolization of the student's is equivalent to the "right symbolization." But for our purposes it is sufficient that the student construct symbolizations according to certain rules of thumb which ensure that different students arrive at a uniform result.

In order to construct derivations and proofs within this symbolic language, we need three inference rules and a new derivation procedure over and above those required for sentential logic. The new derivation procedure is one which permits the student to establish a formula beginning with a universal quantifier as a line of a derivation or proof.

An example of such a derivation is the following:

Derive: (∧x)(F(x) & G(x) → G(x))

Gen:x	OK	
WP	(1)	F(x) & G(x)
1RC	(2)	G(x)
1.2CP	(3)	(F(x) & G(x) → G(x))
Gen:c	(4)	(∧x)(F(x) & G(x) → G(x))

By typing 'Gen:x' the student indicates that he wants to establish as a line a formula of the form (∧x)φ. This derivation procedure is intended to capture the proof technique of informal mathematical practice where one demonstrates that

every object has a certain property by showing that an arbitrary object from the universe of discourse has the property. Consequently, as in the case under consideration, the computer must ensure that the object indicated is an arbitrary one in the appropriate sense by scanning the previous lines of the derivation or proof to make sure that the object indicated is not explicitly mentioned. Thus, if x does not occur free in any antecedent line, the derivation is permitted--in this case, the computer types 'OK.' After the student has constructed the desired formula except for the initial quantifier and its variable, he types Gen:c which tells the computer to type a universal quantifier, the variable occurring in the last instruction of the form 'Gen:_', and the formula immediately preceding; i.e., the computer is told to type out a formula of the form $(x)\varphi$.

The three new inference rules are universal specification (US), existential specification (ES), and existential generalization (EG). The use of these rules is not problem-free. In order not to warrant fallacious inferences, authors of standard texts on mathematical logic usually give a definition in a syntactical metalanguage of the notion of proper substitution. In terms of this notion it is possible to formulate certain restrictions on the rules of inference which enable one to avoid generating fallacies.

Within the context of CAI it is not necessary to formulate these complicated definitions. In analogy to the procedure for determining whether a variable of generalization satisfies certain conditions, the student may "ask" whether a certain application of one of the rules is permitted.

In the case of all of these rules of inference one need be careful not to identify variables. The many ways in which this can happen need not be discussed here as that would take us too far into technical details.

As a final example consider the following:

		Derive: $(y)(F(y) \rightarrow H(y))$
P	(1)	$(x)(F(x) \rightarrow G(x))$
P	(2)	$(z)(G(z) \rightarrow H(z))$
Gen:y	OK	
WP	(3)	$F(y)$
1USy	OK	
	(4)	$F(y) \rightarrow G(y)$
4.3AA	(5)	$G(y)$
2USy	OK	
	(6)	$G(y) \rightarrow H(y)$
6.5AA	(7)	$H(y)$
3.7CP	(8)	$(F(y) \rightarrow H(y))$
Gen:c	(9)	$(y)(F(y) \rightarrow H(y))$

As with the case where a student's input has the form 'Gen:β,' where β is a variable, '_US_,' '_ES_,' and '_EI_' are essentially questions which the computer responds to by typing either OK in case the operation is permitted, or an error message telling the student why the operation cannot be performed. If 'OK' is sent, after the student hits the space bar, the desired result is typed out.

Programming. The major part of our effort was spent on debugging the program. Most of the coding had already been completed, so that when the first version of the time-sharing system became reasonably reliable, we began debugging the input/output functions in the program. This was a necessary first step, because when those functions are working well, the rest of the debugging is simplified. The goal was to get to the point where we could present a simple problem to a student, process his input to the program, and then type information back to him. This meant debugging (a) the interface with the system, (b) the routines that handle curriculum files, search for the right problem, read it in and present it to the student, (c) the routines that collect the student input, convert it to internal notation, and decode it so that the proper rule processor can be called, and (d) all the routines that collect the processed information, convert it from interal notation, and then send it out to the student.

We had continued debugging on the earlier version of the time-sharing system until almost the end of the quarter, waiting until the new and final version had been up and running reliably for a while before trying to run under it. We spent the rest of the quarter re-programming the interface with the system, so that we could begin running under the new version.

D. Second-year Russian Program

The period from July 1, 1969 through September 30, 1969 was spent preparing for the coming academic year and included revisions, changes, and additions to the material.

The interest of Stanford students in computer-based Russian was evidenced at the end of the academic year of 1968-69 by the pre-registration of 69 students; 53 students for the first-year course, and 16 students for the second-year course.

Since at the time of pre-registration we had not announced that in the coming year only the computer-based second-year course would be offered, it is significant to note that of the 30 students who completed the computer-based first-year Russian in the academic year 1968-69, 16 students pre-registered for

computer-based second-year Russian. The 14 students who did not re-register were those who either were no longer at Stanford or who had completed language requirements.

During the two days of registration on September 29 and 30, 1969, for the coming academic year, 75 students registered for computer-based first-year Russian as opposed to 13 students registered for the conventionally taught first-year Russian.

E. Computer-assisted Instruction in Programming: AID

1. The Instructional System and Its Implementation

The instructional system which will be developed consists of three major components: a coding language, a set of computer programs to interpret the coding language, and a set of auxiliary operational programs. The coding language will be a high-level programming language used by curriculum writers for writing programmed lessons in machine-readable form. The coding language, which is independent of any particular implementation on any specific computer, must be defined precisely enough so that it can be implemented on any computer with the necessary components (e.g., teletype or typewriter consoles, random-access memory of sufficient size). Since the coding language may be used by relatively inexperienced personnel, it must be easy to learn and easy to use.

The implementation of the instructional system will require a set of programs that interpret lessons written in the coding language and follow those instructions in presenting lessons to students. The set of interpreter programs must not only interpret coded lessons correctly, but must also satisfy other prerequisites of a good system for computer-assisted instruction. One of the most essential prerequisites is satisfactory response time; the computer must be able to respond to the student in a matter of seconds. The implementation must provide also for sufficient storage space for large numbers of coded lessons. Since massive amounts of curriculum will be prepared, the storage of coded lessons must be economical, in terms of amount of disk space used, and at the same time, must provide rapid access to such material to satisfy the requirement of fast response time. The amount of computing time and amount of core memory needed also must be held to a minimum to provide an economic implementation.

In addition to the programs needed to interpret the coded lessons, there must also be available a set of programs for various operational purposes. For example, a simple method of enrolling students is needed. Also, it is desirable to have reports on the progress of specified students or groups of students, reports on the amount of use of the system, etc. These auxiliary programs, while not inherently necessary, help provide a smoothly running large-scale operation.

2. Design of Revised System

A revision of all parts of the instructional system was planned in detail. The revision was based on six months' experience with the preliminary version; suggestions were contributed by students, coders, writers, and machine operators.

Revision of curriculum. During the preliminary test of the lessons, sufficient information was gathered to warrant revising the first fifteen lessons. A few changes were made in the outline and numerous small changes are being made in problems. The course as now planned will use three strands: one for lessons, one for summaries of lessons, and one for reviews, with the possibility of adding several smaller strands, such as additional lessons in algebra and trigonometry, at a later time.

The work of revising and recoding the first twenty-five lessons is about half done. Lessons 26 to 38 are written and coded. Lessons 39 to 50 have not yet been written and no debugging has been done. About two months of full-time work is needed to complete the revision of the curriculum.

A revision of the student manual is also underway and will be completed within a few weeks.

Design of the system. The instructional system has been designed to allow considerable flexibility in curriculum design. A course may be partitioned into a number of "strands," each of which may contain many lessons. Thus, a course in history could be written with one strand for the history of Asia, one for the history of Europe, etc. In any strand, the lessons are numbered 1, 2, 3, etc., so that each strand may constitute a separate sub-course with no necessary relation between the content of various strands. On the other hand, a curriculum designer may wish to use one strand for the basic required lessons, another for enrichment material, and still a third strand for remedial lessons. In such a case, there might be a close relationship between the content of similarly numbered lessons on different strands, e.g., the third lesson on the remedial strand could be for students who had difficulty with the third lesson on the main strand.

The current position on each strand for each student is continuously updated by the system and is saved from day to day so that the student may start each day from the last problem he was working on.

Within each strand, there is a sequence of lessons containing any number of problems. The basic branching structure of the system provides for linear branching from one problem to the next after a correct response is given and an automatic branch to the beginning of the next lesson after a lesson is completed. (An incorrect response to a problem simply causes a loop within the same problem.) There is no implicit branching between strands.

Although the only branching which is done automatically by the system is a simple skip to the next problem in the sequence, extremely complex branching structures may be used in a given course by using the branching commands provided in the coding language. These branching commands are conditional upon either correct or incorrect responses from the student and can cause a branch to any problem in any lesson in any strand. In addition, the branching commands can be used as "subroutine returns" to return a student to his current position on another strand; i.e., if no lesson and problem numbers are specified in the branch command, the student's current position in the specified strand will be taken as the specification.

The flexibility of the branching structure may be enhanced in yet another way. If desired by the curriculum designer, the student may use "control keys" to alter his own sequence of instruction. The curriculum designer can specify any key on the teletype as a "go" key; when the student types that key, the program requests the number of the lesson and problem he wants to do next. There may also be a "next" key, which causes a skip to the next problem in sequence, without requiring the student to specify lesson and problem numbers. The "next" key, like all other control keys, must be specified by the curriculum designer before it can be used by students.

Several other student control keys are available. Use of the "hint" key displays additional explanatory text, if such text was provided by the curriculum writer. Any number of hints may be provided for each problem and are given to the student in the order coded in the problem. The "tell" key requires the computer to give the correct answer to the current problem, again provided that such an answer was coded by the curriculum writer. For both the "hint" and "tell" routines, there are appropriate default messages which will

be typed if the curriculum writer did not supply a message, e.g., "There are no hints for this problem," "No answer was written."

By appropriate definitions of the student control features, a curriculum designer can build a course which allows for maximum student control, or he can design a course in which the branching structure is tightly controlled by the program.

In order to provide highly individualized programmed lessons, there must be non-trivial routines for analyzing student responses and performing appropriate actions contingent upon the results of such analyses. Analysis routines must be highly differential, so that specific errors can be isolated and appropriate remedial material presented. A simple correct-incorrect classification of responses is insufficient for an individualized, tutorial system of teaching. There are twelve basic analysis routines: EXACT, KW, EQ, MC, TRUE, YES, and their negations NOTEXACT, NOTKW, NOTEQ, NOTMC, FALSE and NO. The EXACT routine checks the student response for an exact character-by-character match with a coded text string; KW (key word) checks for the occurrence of a coded key word; TRUE checks for a response of "TRUE" or "T"; the MC (multiple-choice) routine is used for multiple-choice problems in which several choices are correct (a correct response may be a list of all correct choices, or a list of a minimum number of correct choices depending upon how the MC command is used by the coder); the EQ routine checks for a number within a range of numbers, as specified in the coding, or checks for equality with a single number, also as specified in the coding.

The basic analysis routines not only check on the correctness of a student response, they also check on the form of the student response. For example, the EQ routine accepts as a response any number in integer form, decimal form, or scientific notation; any response which is not in an acceptable form (e.g., a response of the word "four") elicits an error-in-form message: ERROR IN FORM: PLEASE TYPE A NUMBER. Another routine which differentiates between correctly formed and incorrectly formed responses as well as between correct and incorrect responses is TRUE. Either TRUE or T is a correct answer, and either FALSE or F is an incorrect answer. Any other response from the student elicits an error-in-form message: PLEASE ANSWER TRUE OR FALSE. Most other analysis routines (YES, MC, etc.) also contain error-in-form subroutines.

Complex analyses of student responses may be made by using simple Boolean combinations of the basic analysis commands. For example, the coder can specify a check for a number that is between 1 and 10 but is not equal to either 5 or 5.5, by using appropriate combinations of EQ and NOTEQ commands.

Since most of the action performed by the analysis routines is internal, i.e., with no action visible to the student, there are also commands that cause coded messages to be relayed to the student, appropriate branching to take place, etc. These commands, called "action commands," are all contingent upon the results of the analyses performed by the analysis commands, i.e., the actions are contingent upon the correctness of the student response.

A detailed description of the design of the system has been written as a part of the Coders' Manual for the PDP-10 implementation.

Revision of computer programs. The revised instructional system will be implemented on the PDP-10, a larger, faster machine than the PDP-1. The major programs, the lesson processor and the lesson interpreter, are now completely written and debugged for use by single users (i.e., the only time sharing which is done is handled by the time-sharing monitor, not by the program itself, so that multiple users of the program require multiple copies). The conversion to a time-sharing system will be accomplished within a few weeks.

No changes have been made in the basic ideas of any part of the system; the main thrust of the revision is more in the nature of an extension. Although none of the basic ideas have changed, several undesirable features were detected in the period that the preliminary system has been in use. Most of these were unanticipated (the reason for preliminary testing is, of course, to uncover unanticipated errors in the design). For example, the lesson interpreter continuously updated its record of the student's postion in the course, but did not save the record from day to day without a specific request from the machine operator. This proved to be a minor source of annoyance during the period when several different operators (some quite inexperienced) were using the system. Another inconvenience for machine operators that occurred as they loaded the system was the requirement to specify the desired lesson file by number and name. A simpler operational procedure was to specify a course numer and to let the program determine the number and name of the appropriate lesson file. Both of the above features have been changed in the revision.

In the preliminary version of the lesson processor, there was a limit on the amount of lesson code which could be processed at one time--the limit was imposed by the amount of available core storage. This limit was large enough so that most lessons could be processed in their entirety but the coders had to be continually aware of the limit and had to break long lessons into several segments for processing. In the new version, no limit is made on the amount of code which can be processed at one time.

In coding a lesson, the coder was required to list, at the beginning of each lesson, the numbers of all the problems in the lesson. These numbers had to be sequential and had to correspond exactly to numbers attached to the problems in the lesson. Not only was this a time-consuming procedure, it also led to unnecessary coding errors. In the revised system, problems are numbered automatically, simplifying the coder's task as well as decreasing the error rate.

All changes discussed so far are minor, but of increasing importance when large amounts of curriculum material are produced and large numbers of students accommodated. A smoothly running system should allow as little room for operator and coder errors as is feasible.

As mentioned before, the main direction of the revision was to extend the ideas already incorporated in the system. The three major changes in the instructional system were:

1. Revision of the lesson interpreter to allow time-shared use of the system by large numbers of students.
2. Addition of a multiple-strand structure for the curriculum.
3. Addition of multiple-hint capabilities.

Extensive changes are planned for the coding language, the curriculum, the interpreter program, and the processor program.

The coding language is being extended to include about twice as many basic commands as were available in the preliminary version, the format has been simplified, and macro capabilities were added.

A detailed description of the coding language and how to use it has been written as part of the Coders' Manual. The Coders' Manual also gives a complete description of all programs and how to operate them, including instructions for using other necessary systems programs such as the text editor, the assembler, and the loader.

The lesson processor. The new lesson processor is similar to the old one; it is also a two-stage processor, the first stage being one of the PDP-10 assemblers. Since the PDP-10 has a macro-assembler, full advantage has been taken of the macro capabilities. The processor consists almost entirely of macro definitions of the op codes used in the coding language, plus a very short load routine which stores the processed lessons on a disk file (the processor is essentially a zero-length program). The coder is also allowed the advantages of a macro assembler; judicious use of macros can reduce coding time significantly.

In the preliminary version of the system, the translation of text to teletype code (which must be done before messages can be typed on a teletype) was done by the lesson interpreter. This is a somewhat inefficient arrangement since the translation must be done for each student using a problem, and, while not a time-consuming operation, it does contribute to an increase in response time. In the revision, the translation to teletype code is performed by the lesson processor so that it is done only once per problem rather than once for each use of the problem.

The lesson interpreter. The new interpreter is written as a re-entrant time-sharing program using about 5k words (36 bit) of core plus 1k for each of the students concurrently taking lessons. The program is written in one of the assembler languages for the PDP-10.

As with the preliminary version, great care has been taken to ensure fast response time and economical use of core and disk storage. Routines for detecting and compensating for coding errors have been incorporated. In a similar fashion, unexpected responses from students are not allowed to cause errors in the program.

Two changes in the lesson interpreter serve to make the instructional system much more versatile than the preliminary version. One of these changes is the provision of variable student control keys. That is, the actual keys used by the student as a "hint" key, a "tell" key, etc., are specified by the curriculum designer, and may, in fact, be left undefined if students are not to be allowed the use of some control keys.

Another change, which is in the same spirit as the variable control keys, is the provision for variable standard messages. Standard messages such as NO HINT WAS WRITTEN, and ERROR IN FORM: PLEASE TYPE A NUMBER were previously fixed by the lesson interpreter. In the new version, such standard messages are defined

by a coder when a new course is started and may be changed as many times as desired without affecting the messages used in a different course.

The device of allowing variable control keys and standard messages allows the program to be used simultaneously by students in many different courses, without requiring the designers of the course to agree in detail on control keys and standard messages.

Both the lesson processor and the lesson interpreter are described in some detail in the Coders' Manual and are also documented separately.

Documentation. The main document for the instructional system is the Coders' Manual which contains a complete description of the instructional system, an introduction to the coding language, reference sections on the coding language, a description of all programs, and instructions for operating them. The manual, which contains about 100 pages, is written for use by curriculum designers, writers, and coders who are unfamiliar with computers and programming. The manual is written so as to separate those sections that pertain to a particular implementation; thus, the first part of the manual, which includes a description of the system and the instructions on coding, could be used for other implementations.

The manual is now complete except for a section on operating the enrollment program and an index.

In addition to the documentaion in the Coders' Manual, all programs are also documented internally. The Coders' Manual, together with a set of program listings, would constitute sufficient specification for another implementation.

F. Computer-assisted Instruction in Programming: SIMPER and LOGO

1. Background of the Project

In February, 1969, a course in elementary computer programming was introduced at Woodrow Wilson High School in San Francisco. Six classes of 15 students each received instruction via teletypewriters linked to the PDP-1 computer at the Institute. Two adult education classes and an afternoon "enrichment" laboratory for upper elementary and junior high school students also used the equipment and instructional materials.

Two programming languages were taught, both especially designed for instructional purposes. One was SIMPER, a pseudo-assembly language designed by Paul Lorton of Stanford University. The other was LOGO, a list-processing

language designed by and used with the permission of Bolt, Beranek, and Newman, Inc. of Maynard, Massachusetts. There were 38 SIMPER lessons and 19 LOGO lessons.

2. Curriculum Revisions

In July, it was believed that both SIMPER and LOGO would be taught on the PDP-10 computer in September, 1969. Also, an advanced course to teach AID (a course described in Part E of this report) would be offered on the PDP-10. The lesson-control program for these courses is also described in detail in Part E. The first 20 SIMPER lessons were revised using teachers' and students' comments in a form suitable for use with the new system, as were the first 20 LOGO lessons.

As work on the PDP-10 programs continued on the general systems work, the lesson-control program, and implementation of the three programming languages, it became clear that a deadline of September 4 was not realistic. These considerations led to a revised plan: to continue to use the PDP-1 computer for the fall semester for SIMPER and LOGO. Since LOGO would be extended from 19 to 49 lessons and extensively revised, second-year students would begin with a review of LOGO, then continue with new LOGO lessons, going on to work in the advanced course on AID about the end of December. These advanced students would then continue the AID course in the second semester.

To implement this plan, the LOGO lessons were recoded for the PDP-1. By September 30, the first 30 LOGO lessons had been revised and coded, and the next 19 lessons planned. The outline of these lessons appears in Table 8. Homework assignments and tests for these lessons were also written and coded.

A student manual was prepared that included discussions of computers, programming languages, and computer-assisted instruction, as well as detailed descriptions of the teaching program and LOGO, and a glossary of terms used in the LOGO course. A teachers' manual was also prepared which included the student manual, homework assignments, answers to homework and tests, and suggestions for effective implementation of the course.

3. The Classroom

On September 4, the LOGO course was begun in eight classes. A few weeks later the two adult classes resumed work, as did the enrichment laboratory. As during last year, students and school personnel have been enthusiastic and extremely helpful in all work associated with the project. Several students came to school early, usedlunch periods, and/or remained afterschool to gain more time to work on the course.

TABLE 8

Outline of LOGO Lessons, Lessons 1 to 49

Lesson	Description
1	Introduction to the Teaching Program.
2	Introduction to LOGO; the instruction PRINT.
3	The instructions WORD and SENTENCE.
4	The instructions FIRST and BUTFIRST.
5	Review: Lessons 2 to 4.
6	Review: Lesson 1.
7	The instructions LAST and BUTLAST.
8	Introducing compound commands.
9	More about compound commands.
10	Signing on to LOGO.
11	Review: Lessons 7 to 10.
12	Review: Lessons 2 to 10.
13	Arithmetic instructions.
14	Negative numbers.
15	The instruction CALL.
16	Using CALL with other commands.
17	Review: Lessons 13 and 14.
18	Review: Lessons 15 and 16.
19	The instruction SAME.
20	The instructions WORD?, NUMBER?, and SENTENCE?.
21	The instruction GTR?.
22	Introducing functions.
23	Review: LOGO tests.
24	Review: Functions.
25	Practice writing functions.
26	Functions with more than one argument.
27	Editing functions.
28	The instruction GO TO.
29	Equivalent functions.
30	Review: Lessons 25 to 29.

TABLE 8 (continued)

Lesson	Description
31	Review: Lessons 1 to 29.
32	An astrological function.
33	Introducing recursive functions.
34	Writing recursive functions.
35	String searches using recursive functions.
36	Review: Recursion.
37	Review: Lessons 1 to 35.
38	Writing functions to do "Piglatin."
39	More recursive functions.
40	Advanced recursive functions.
41	Writing functions about sets.
42	Review: Recursion.
43	Review: Lessons 1 to 41.
44	Writing functions involving decimals.
45	The instruction BOTH.
46	The instruction EITHER.
47	Loops.
48	Counters.
49	Review of the course.

C. Basic Research in CAI

This is the first progress report covering work begun under the grant for basic research in CAI. The report therefore mainly deals with work that has just begun or is still in the planning stage. A description of the work is given under the main categories of activities planned through the coming year.

Dialogue programs. Three students, Adele Goldberg, James Moloney, and Roulette Smith, are writing dissertations on dialogue programs under the direction of Professor Patrick Suppes. Miss Goldberg, who is a doctoral student at the University of Chicago, is working on a program that will be able to help students proving theorems in algebra, by attempting to look ahead at the steps needed to complete the proofs begun by the student. The central problem is to embody in a program a proof procedure close to the kinds of proofs students will ordinarily give. Mr. Moloney is working on a related problem. In his case, he is dealing with proofs in algebra restricted to identities. He is attempting to impose a metric on the distance between each line of proof given by the student and the desired goal, so that the program can, by constructing a proof itself, extend the work of the student. The program looks for a step that diminishes the distance between the work done thus far by the student and the intended final result. Mr. Smith is writing a program that is meant to simulate the behavior of teachers in teaching elementary sentential logic. In his case, he will build up a store of proofs given by students and use these rather than a generative procedure. He is also planning to introduce a dialogue of comments that are meant to be tailored after the kind of comments ordinarily given by teachers.

Speech recognition. During this first period of activity, the main work has been establishing close communication with Professor D. Rajagopal Reddy's group, which has been moving from Stanford to Carnegie-Mellon. Close cooperation with Professor Reddy and his group working on basic problems of speech recognition is planned for the future. We are just beginning the first experiments of attempting to recognize a vocabulary of twelve words, the ten digits, yes, and no, spoken by young children over telephone lines. The first step is to use Reddy's present program, now running on the PDP-10 of the Artificial Intelligence Project at Stanford. On the basis of these initial experiments, we will then decide what kind of software and hardware system to put on our own PDP-10 system for more extensive experimentation.

Mathematical models of learning and performance. The main effort in this direction during the period covered by this report has been directed toward developing more detailed automaton models of student performance in elementary mathematics. In the original theoretical work in this direction, we restricted ourselves to automata that dealt just with answers being correct or incorrect, i.e., if an error was made it was simply an error according to an error distribution and not according to some wrong conceptual scheme. In many cases, students make errors of a systematic sort, as everyone who has taught any level of mathematics is well aware, and what we are currently trying to do is to embody a more elaborate scheme in our automaton models to take account of these phenomena. We are in the process of deciding what data to collect in order to test the theoretical ideas. During the course of the year we hope to be able to complete some fairly substantial empirical tests of the theoretical models we have been developing in this area. More detailed reports of these tests will be given in later progress reports.

Computer-generated speech. Work on a digitized audio system began in 1968 after we decided that any straightforward analog system would be unworkable. Previously, we had worked on the CROW system which had a vocabulary recorded on a wide magnetic tape. That system was sold to the University of Pittsburgh.

By June, 1969, we had a working digital system that required 72,000 bits for each second of audio, was noisy, and had a very limited frequency response. The system was only able to pass frequencies below 430 hz at full amplitude. The maximum amplitude for frequencies fell off at 6 db/octave above that frequency. (Note: All db figures in this report are power figures, i.e., db = 20*log (V1/V2.)

We have changed the bit rate to 36,000 hz, eliminated all but the quantization noise, and increased the critical frequency from 430 hz to 850 hz.

We have further reworked the speech editor "SPEDIT" to accommodate these changes and have added various simple features to the program. The new editor is called "SPEECH" and lives on the PDP-10. Two other versions exist: "FOPS" edits and plays, but does not record; "SHORT" plays only.

These programs are for the series 3.18 monitor and are being converted to the series 4 monitor.

Description of the single-bit audio follower system. The single-bit audio follower system works by only recording, in digital form, the changes in amplitude of the analog audio waveform. The dynamic cardiod microphone in booth F transduces a person's speech into an analog voltage. This voltage is amplified by the Bogen model RTP-1 pre-amplifier. The line level output (600 ohms, 1 milliwatt) of this pre-amp is filtered by two Krohn-Hite filters. One filters out all frequencies below 300 hz and the other filters out all frequencies above 300 hz. This filtered input wave is buffered by a unity gain amplifier in the rack. Its output voltage is compared with that of an integrator. If the input amplitude is larger than the output of the integrator at the sampling time, then a ground is applied to the integrator for the duration of the sampling period. If the input amplitude is smaller than the output of the integrator, a positive voltage is applied to the integrator. This circuit is called the feedback DAC (digital-to-analog converter). The feedback DAC is a series 10k resistor and a parallel .47 mfarad capacitor followed by an amplifier with a voltage gain of -22. Thus, a positive voltage applied to the input will cause the output of the amplifier to ramp downwards. A ground applied to the input will cause the output to ramp upwards. A 36,000 cycle/sec. sampling clock is used. A bias voltage is needed to balance out all the offset voltages of the amplifier chips used.

There are two important parameters in the design of this audio system. The first is the slope of the ramp produced by the feedback DAC. The second is the sampling frequency or bit rate. One picks the rc time constant of the integrator to be longer than the time constants of the audio's lowest frequency (i.e., 300 hz). This allows any noise picked up in the integration process to gradually bleed out and at the same time allows full frequency response. One then sets the gain of amplifier to give a particular slope. This slope determines the highest frequency that the analog-to-digital converter can follow at maximum amplitude. This system's slope allows it to follow all frequencies below 800 hz at full amplitude. It can follow lower amplitude high frequencies since the maximum that can be followed falls off at 6 db per octave. The sampling rate and the slope jointly determine the amount of noise in the system. Thus, a fast slope increases the frequency response, but also increases the noise, and a fast smapling rate will decrease the noise, but increase the channel bandwidth requirements.

The bit stream that is input to the feedback DAC is also input to a high-speed line unit (HSL). Normally, the HSL ignores this bit stream. However, if the record button in booth F is pushed, the clock also goes to the HSL and gates in the bit stream. The HSL packs the bit stream into 18-bit words and places these words in core. The program SPEECH, if it is running, unpacks the 18-bit words and repacks them as 36-bit words. The program can then edit, play, and file the sound on the disk. The program can also retrieve old sounds from the disk. Once on the disk, other programs, such as the drill driver, can get these sounds and play them.

Playing the sound is accomplished by placing the sound in core and placing a pointer to that buffer where the audio multiplexor will see it. The audio mux will see the pointer, increment it by one, and then access the word. This word will be placed in a 36-bit shift register. From the shift register, one bit at a time will be shifted to an output DAC. At present, there are 72 (octal) channels each with a shift register and DAC. The output DAC is much like the feedback DAC except its amplifier has a gain of -44 and it has an rc filter on the output (f = 3000 hz).

Acquisition of reading skills. An experimental program in this area is just beginning under Professor Richard C. Atkinson and will be reported in more detail in the next Quarterly Report. During the period covered by this report, the main emphasis has been on getting the current PDP-10 system operational, including digitized audio, in order to create a framework within which experiments could be conducted.

H. Stanford PDP-1/PDP-10 System

1. Hardware

After the end of summer school, the PDP-10 was modified by DEC engineers to install the second set of relocation and protection registers in the central processor. These were necessary to run the re-entrant programs of the new monitor system. The work was completed in good time, and no problems have been experienced with the new features.

During this reporting period, one of the two IBM 2314 disk units was removed from the system because of reduced storage requirements after the Mississippi and Kentucky student terminals were detached from the system.

An analog-to-digital input channel was attached to the PDP-10 through one of the units of the high-speed data communications multiplexor. This is used for audio recording and permits the input of digitized sounds directly into the PDP-10 memory, instead of passing the data through the PDP-1, which had been the previous technique.

A process of investigation and experimentation was begun to improve the quality of the speech output by the digitized audio system. The analog-to-digital channel mentioned above was part of this program to obtain more understanding and control of the data at each stage from recording, processing, and storage to the distribution and conversion back to audible speech.

Work was started to convert much of the local communications system from single lines to time-division multiplex arrangements permitting several terminals to share a single line. Due to delays in the delivery of some of the equipment, it was not possible to complete the change-over during this period.

The magnetic-tape audio units used by the Russian course were the subject of some redesign during this period to improve their audio quality and control reliability.

There were continuing problems with minor circuit failures and intermittent faults in the PDP-10 memory system, but the remainder of the central facilities had good reliability.

2. Software

The major effort during this period was on the adaptation of the DEC Series 4 time-sharing monitor for use on the Stanford hardware. An important advantage of this type monitor to the operations is that the system programs insulate users from one another more efficiently than the drill-driver system. When using an experimental program, for example, even a serious programming error will affect only those users who reach that particular section of code, while others may continue. It is also possible to correct or improve programs by making a new version available without interfering with users who have already begun working with the older version.

As the system is used for a wider range of courses, it is also important that terminals may be used for as many of these as possible without intervention from an operator to "detach" or "attach" the unit with respect to particular programs. The new monitor will permit the student a full choice of courses for

which he is registered, while excluding him from others. In a similar way, it will no longer be necessary for a driver program to be "brought up" at a particular time for it to be available. It should be possible to get lessons or reports at any time the system is in operation.

The unique hardware arrangement of the Stanford system required many changes to the monitor, but the most important of these were completed during this period. These included communication with the PDP-1 and the use of the Philco displays as terminals. Code for the IBM 2314 disk unit was written, including a technique for using individual packs that does not currently appear in DEC software.

Utility programs were written or modified for the new system, and memory and other test programs were developed to assist with hardware maintenance.

II. Activities Planned for the Next Reporting Period

A. Drill-and-practice Mathematics Program

The revision of the remedial college-level mathematics will begin during the next period. A sufficient number of students have now run on the preliminary version of the program to indicate the nature of the revisions needed. In particular, revisions will include more semi-programmed instructional segments to introduce key points or to help students in identified problem areas.

Format problems continue to be of concern. It is difficult to format many standard algebraic expressions on a Model-33 teletype. Several approaches have been tried. One which appears to offer the most promise is to use a more conversational instructional approach while directing the student through each part of the expression and then to recombine the parts into an equation for simplification in the final steps.

The first quarter's work will be revised and the second quarter will be completed during this period.

The problem-solving program which has been at a standstill will receive more attention. A large number of items (2,000 to 3,000) have been written. Many need revision and classification into equivalence classes. Problem classification itself is not a simple task. Little is known about the variables that contribute to difficulty and confusion in arithmetic word problems. The literature is being searched for suggestions as to which variables appear to have the most influence at each grade level and in each conceptual area across grade levels.

Programs developed at Stanford and elsewhere will be used with adults interested in passing the G.E.D. test or obtaining a high-school diploma. Four teletype terminals will be in operation with the Stanford Medical School staff in a program designed to upgrade the education of staff members, particularly those who are members of minority groups. This program is being sponsored by the Minority Relations Department at the Stanford Medical School.

Work on the strands mathematics program will continue through the year. The program eventually will be extended upward to include grades 7 and 8.

B. Drill-and-practice Reading Program

Routines for on-line monitoring of reading students and for altering individual and class restart parameters will be completed for the 'Series 4' monitor. Attention will again be directed to the development of meaningful data recording, collection, and analysis. Plans are being made to re-record the audio during this quarter for greater intelligibility. Every effort will be made to solicit suggestions from classroom teachers whose students are involved in the CAI reading program to further implement the curriculum and individualize the reporting procedures to improve the program's effectiveness as a teaching tool. We will also be involved in the development of the language arts strand and in making the necessary revisions in the Teacher's Manual.

C. Logic and Algebra Program

The debugging of the logic and algebra program, which is necessary after the extensive revision and reprogramming, is expected to be completed. This program includes the new counterexample mode.

Having completed the recoding, including the systematic replacement of the sentential connectives by their English counterparts, the curriculum is ready for debugging. It is expected that this will be accomplished by the next reporting period.

D. Computer-assisted Instruction in Programming: AID

During the next three months, the lesson interpreter program will be revised to make it a re-entrant time-sharing program, thus completing the set of essential programs. The Coders' Manual will be completed also. The major effort will be in the revision and extension of the curriculum and some effort will be put into designing desirable auxiliary programs for the instructional system.

E. Computer-assisted Instruction in Programming: SIMPER and LOGO

Next quarter LOGO lessons 31 through 49, together with associated homework assignments and tests, will be written and coded. Material on all LOGO lessons will be gathered to aid in revisions. We plan to offer the LOGO course in autumn, 1970, when it will be available on the PDP-10 computer. The language itself is being written for the PDP-10 by its original designers, Bolt, Beranek, and Newman, Inc. of Maynard, Massachusetts, and should be debugged sometime in early 1970. Our revised lessons will take advantage of new commands in the PDP-10 version of LOGO and will incorporate suggestions of teachers and students for improving current lessons.

Also, preparation of a course in BASIC will be started. The BASIC course will be offered beginning January 27, 1970 and will run on the PDP-10 computer. The lesson-control program and the coding language that will be used is in preparation and is described in part E of this report.

F. Stanford PDP-1/PDP-10 System

No hardware changes are planned for the next reporting period. Work will be completed on the multiplexed communication line system. Delivery is expected of a DEC 6801 communications unit which will expand the capacity of the system for local terminals and those using the multiplexed lines. It is not planned to bring this unit into use during the period. The 1301 disk unit on the PDP-1 will be released if the necessary software can be completed for the PDP-1 to use space on the 2314.

Software work should see the operation of the Series 4 monitor for instructional use by students.

III. Dissemination

A. Lectures

Atkinson, R. C. Chairman, Colloquium on Memory at the XIX International Congress of Psychology, London, July 27-August 2, 1969.

Jerman, M. Computer-assisted instruction in elementary and secondary mathematics. Lecture presented to Shell Merit Fellows, Stanford University, Stanford, July 11, 1969.

Jerman, M. Computer-assisted instruction. Lecture presented at The Workshop on Innovations in Instructional Techniques, Colorado College, Colorado Springs, July 21, 1969.

Jerman, M. The Stanford drill-and-practice program in elementary mathematics. Lecture and demonstration presented at The Workshop on Innovations in Instructional Techniques, Colorado College, Colorado Springs, July 21, 1969.

Jerman, M. The philosophy and structure of available CAI programs. Lecture presented at The Workshop on Innovations in Instructional Techniques, Colorado College, Colorado Springs, July 22, 1969.

Jerman, M. Computer-assisted instruction in mathematics. Lecture presented to NSF Summer Institute participants, Stanford University, Stanford, July 30, 1969.

Jerman, M. Drill-and-practice in elementary mathematics. Lecture presented to selected mathematics teachers, St. Paul City Schools, St. Paul, Minnesota, September 17, 1969.

Jerman, M. Research studies in individualized technology. Lecture presented at the Teacher Institute on Technology in Education, Milwaukee Diocese, Bruce Hall, Milwaukee, September 18, 1969.

Jerman, M. Current progress in technological education. Lecture presented at the Teacher Institute on Technology in Education, Milwaukee Diocese, Bruce Hall, Milwaukee, September 18, 1969.

Jerman, M. Computer-assisted instruction: the state of the art. Two lectures presented to supervisors and curriculum specialists of Detroit Public Schools, Detroit, September 19, 1969.

Suppes, P. Research in individualized instruction--implications for facilities. Lecture presented at 19th Annual Summer School Planning Institute, Stanford, California, July 7, 1969.

Suppes, P. Course and curriculum development at the undergraduate level. Informal statement and discussion at 12th Meeting of the National Science Foundation's Advisory Committee for Science Education, Harvey Mudd College, Claremont, California, July 11, 1969.

Suppes, P. Discussant, Symposium on Deductive Thought, XIX International Congress of Psychology, London, England, July 29, 1969.

Suppes, P. Chairman, Symposium on Computer-assisted Instruction, XIX International Congress of Psychology, London, England, July 30, 1969.

Suppes, P. Stimulus-response theory of finite automata and language learning. Lecture presented at University of Minnesota, Minneapolis, August 15, 1969.

Suppes, P. Finite automata and language learning. Lecture presented at Mathematical Psychology Meetings, University of Michigan, Ann Arbor, August 28, 1969.

Suppes, P. Technological innovations. Lecture presented at symposium on early learning and compensatory education, American Psychological Assocation, Washington, D. C., August 31, 1969.

Suppes, P. Seminar with Professors R. Duncan Luce, Amos Tversky, and David Krantz on Basic Research in the Theory of Measurement with Specific Applications to the Behavioral Sciences, Institute for Advanced Study, Princeton, New Jersey, September 1-5, 1969.

B. Publications

Atkinson, R. C. Computer-assisted learning in action. The Proceedings of the National Academy of Sciences, 1969, 63, 588-594.

Atkinson, R. C. Information delay in human learning. Journal of Verb. Learn. and Verb. Behav., 1969, 8, 507-511.

Atkinson, R. C. Models for memory. In F. Bresson and M. de Montmollin (Eds.), Sciences du comportement: La recherche en enseignement programmé. Paris: Dunod, 1969. Pp. 75-92.

Atkinson, R. C., Fruend, R. D., and Brelsford, Jr., J. W. Recognition vs. recall: Storage or retrieval differences? Quarterly Journal of Experimental Psychology, 1969, 21, 214-224.

Atkinson, R. C., and Rundus, D. Rehearsal processes in free recall: A procedure for direct observation. Technical Report No. 149, August 12, 1969, Stanford University, Institute for Mathematical Studies in the Social Sciences.

Jerman, M. Promising developments in computer-assisted instruction. Educational Technology, 1969, 9, 10-18.

Suppes, P. Stimulus-response theory of finite automata. Journal of Mathematical Psychology, 1969, 6, 327-355.

Suppes, P. Stimulus-response theory of automata and tote hierarchies: A reply to Arbib. Psychological Review, 1969, 76, 511-514.

Suppes, P. Nagel's lectures on Dewey's logic. In S. Morgenbesser, P. Suppes, and M. White (Eds.), Philosophy, Science and Method, Essays in Honor of Ernest Nagel. New York: St. Martin's Press, 1969. Pp. 2-25.

Suppes, P. Computer-assisted instruction. An interview with Patrick Suppes. The Education Digest, 1969, 34, 6-8.

Suppes, P. Research for tomorrow's schools: Disciplined inquiry for education. Report of the Committee on Educational Research of the National Academy of Education. London: Macmillan, 1969. (edited jointly with L. J. Cronbach)

Suppes, P., and Jerman, M. A workshop on computer-assisted instruction in elementary mathematics. The Arithmetic Teacher, 1969, 16, 193-197.

Suppes, P., and Jerman, M. Some perspectives on computer-assisted instruction. Educational Media, 1969, 1, 4-7.

Suppes, P., and Jerman, M. Computer-assisted instruction at Stanford. Educational Television International, 1969, 3, 176-179.

Suppes, P., Loftus, E., and Jerman, M. Problem-solving on a computer-based teletype. Educational Studies in Mathematics, 1969, 2, 1-15.

Suppes, P., Meserve, B., and Sears, P. Teacher's Edition for Sets, Numbers, and Systems. New York: Singer, 1969.

Suppes, P., Meserve, B., and Sears, P. Sets, Numbers, and Systems, Book 2. New York: Singer, 1969.

Suppes, P., and Morningstar, M. Computer-assisted instruction. Science, 1969, 166, 343-350.