MEMORANDUM
RM-3879-PR
OCTOBER 1963

TIPL: TEACH INFORMATION PROCESSING LANGUAGE

Robert Dupchak

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-3879-PR
OCTOBER 1963

TIPL: TEACH INFORMATION PROCESSING LANGUAGE

Robert Dupchak

This research is sponsored by the United States Air Force under Project RAND—contract No. AF 49(638)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

PREFACE

TIPL (Teach Information Processing Language) is a computer program which checks the correctness of solutions to problems written in Information Processing Language-V (IPL-V). It is a teaching aid designed to automatically evaluate both the correctness and the efficiency of a student's program.

TIPL was developed at Carnegie Institute of Technology and The RAND Corporation and was used at the 1963 Summer Institute on the Simulation of Cognitive Processes, held at The RAND Corporation. It proved to be a highly useful tool.

This Memorandum is being released in order to make TIPL available wherever IPL-V is being taught, and to provide necessary information to the potential user for modifying the program to fit the particular IPL-V System being implemented. The Memorandum, of course, assumes familiarity with IPL-V.

SUMMARY

TIPL is a system to assist students in learning IPL-V, a list-processing computer language, and is used in conjunction with the problems contained in the Information Processing Language-V Manual.* It accepts as input a student's program and it proceeds to check the correctness of the program.

The first section of this Memorandum is intended for the student. It describes how he must prepare his program deck and what conventions he must observe. The second section describes how the system operates and the manner in which the instructor may modify old problems and add new ones.

Since TIPL is written entirely in IPL-V, it requires only minimal effort to incorporate it into any IPL-V processor. Those interested in obtaining tape copies of the program should write The RAND Corporation for further information. The program is written on IBM tape as a single file of BCD card images.

---

*Newell, Allen (ed.), Information Processing Language-V Manual, 2d ed., Prentice-Hall, Englewood Cliffs, New Jersey, 1964. The problems with which this Memorandum is concerned were not carried in the first edition of the Manual, published in 1961.

CONTENTS

I. STUDENT INSTRUCTIONS

TIPL is a system to assist students in learning IPL. It accepts as input a student's program and it proceeds to test the correctness of the program. It does this by first supplying a set of data for the program to operate on. It then fires the program. When the program terminates, TIPL checks the results. It also checks to see if W0-W9 remained safe over the program and if the program has erased all the temporary data structures it may have created during its execution. In some problems where the student's program is required to take differential action, TIPL will test the student's program on several sets of data.

The TIPL deck is set up in the following order:

1. One Type-9 card.
2. One Type-5 card, P = 4, Q = 0, and SYMB = 2.
3. All Type-2 cards giving only region size.
   You may use any of the 36 possible regions except the period region and the dollar region. No region may have more than 101 symbols.
4. One Type-T card with NAME = problem number.
   Your name must appear in the comments field of this card.
5. Sets of data and routines, in any order.
6. A final Type-5 card with .0 for SYMB.

A complete example of a correct TIPL solution to Problem 12* is given in Fig. 1.

In your program you may use any and all of the IPL diagnostic aids. You are encouraged to trace your first few programs. Also, you may want to print the inputs to your program and any outputs it generates. If you create any temporary data structures, you may want to print these at various points in your program.

In each problem TIPL will impose a cycle limit on your program. This limit is always far greater than any correct program will require. This limit is imposed to prevent any program caught in an endless loop from using up valuable computer time doing nothing. If your program gets caught in such a loop, TIPL will print an error message and then give you a post mortem. By looking at this post mortem you should be able to tell where the loop occurred.

Problems 10 through 75 may be run using TIPL. In all problems TIPL supplies all HO inputs. In some problems it also supplies various data structures and subroutines;

---

*All problems referred to in this Memorandum are those found in the IPL-V Manual, Ibid.

	COMMENTS	TYPE	NAME	SIGN	PQ	SYMB	LINK	COMMENTS	I.D.
00000	00001111111111222222222233333333334	4	444444	4	45	555555	555666	6666666777	77777778
12345	67890123456789012345678901234567890	1	234567	8	90	123456	789012	3456789012	34567890
		9							
	RØBERT DUPCHAK PRØBLEM 12	5			40	2			
	DEFINE REGIØNS TØ BE USED	2	A				2	DUPCHAK	P32 000
		2	B				2	DUPCHAK	P32 010
		2	P				33	DUPCHAK	P32 020
		2	T				2	DUPCHAK	P32 030
		2	X				7	DUPCHAK	P32 040
	RØBERT DUPCHAK	T	12					DUPCHAK	P32 050
		5						DUPCHAK	P32 060
	PRINT LIST X6		P32		10	X6		DUPCHAK	P32 070
						J151		DUPCHAK	P32 080
	LØCATE A1 ØN LIST X6				10	X6		DUPCHAK	P32 090
					10	A1		DUPCHAK	P32 100
						J62		DUPCHAK	P32 110
	STØRE LØCATION IN W0					J50		DUPCHAK	P32 120
	REPLACE A1 WITH B1				10	B1		DUPCHAK	P32 130
					21	W0		DUPCHAK	P32 140
	PØP UP W0					J30		DUPCHAK	P32 150
	REMØVE T1 FRØM LIST X6				10	X6		DUPCHAK	P32 160
					10	T1		DUPCHAK	P32 170
						J69		DUPCHAK	P32 180
	PRINT LIST X6 AGAIN				10	X6		DUPCHAK	P32 190
						J151	0	DUPCHAK	P32 200
	FINAL TYPE 5 KICKØFF CARD	5				.0			
12345	67890123456789012345678901234567890	1	234567	8	90	123456	789012	3456789012	34567890

Fig. 1--Example of a Correct TIPL Solution to Problem 12

these are listed below.

PROBLEM	TIPL SUPPLIES
12	List X6
13	List X7
14	List X8
31	Subroutine P78
37	Subroutine P88
47	Empty described lists B1 and B2
49	List L12 and empty described list L13
51	List L14
52	List L14
53	Lists L12 and X12
54	List L12
55	List L12
56	Tree L10
60	Subroutine Q11
62	Subroutine Q13
67	Subroutine Q18
68	Subroutine Q17
69	Subroutine R81 and list L17
70	List L18
72	Print line X1 and a card for J180 to read
73	Print line X1 and cards for J180 to read
74	Print line X1 and cards for J180 to read

II. INFORMATION FOR INSTRUCTORS

OPERATION

TIPL is a program written in IPL; therefore it can be used on any IPL computer. As written it requires about 13,000 words of core storage, but it is possible to reduce this to 4000 words and even less by modifications described below. The program requires a KICKOFF DECK which should call IPL and specify an IPL external tape number two. The construction of the KICKOFF DECK will depend on the machine system used. This KICKOFF DECK is used to assemble TIPL, to run students' solutions, and to add and modify problems.

The TIPL program deck (about 8800 cards) is preceded by the KICKOFF DECK. The program deck will then assemble and do J166--SAVE ON UNIT (2) FOR RESTART. This RESTART tape is used to run student problems, modify old problems, and add new problems.

The students' programs are batched and preceded by the KICKOFF DECK. The RESTART tape should be mounted with file protect. TIPL will then read, assemble, and test each student's program.

The second card in the student's deck (Type-5, P = 4, Q = 0, and SYMB = 2) causes a new copy of TIPL to be read

into core from the RESTART tape. TIPL then reads in (using J180's) the student's Type-2 cards and it modifies the region control words* according to the specifications on these cards. When it hits a Type-T card it notes the problem number and it saves this card image. TIPL then does J165--LOAD ROUTINES AND DATA--and this causes the student's routine(s) and data to be loaded. This terminates PHASE ONE of the TIPL program. Note that because the region control words have been modified, any attempt by the student to load undeclared symbols will cause the IPL loader (J165) to print an "UNDEFINED SYMBOL" error message.

The last card in the student's deck is a final Type-5 card that kicks off to TIPL, PHASE TWO. Corresponding to each problem number there is a PROBLEM STRUCTURE. The name of the PROBLEM STRUCTURE is a period symbol whose subscript is ten times the problem number. Thus, the PROBLEM STRUCTURE for Problem 24 is .240. The format for PROBLEM STRUCTURES will be described later. The information in a PROBLEM STRUCTURE gives a series of PROBLEM PARAMETERS and a list of DATA SETS.

In the PROBLEM PARAMETERS is the name of the STUDENT'S

*See §18.0 in the IPL-V Manual.

ROUTINE, a CHECKING ROUTINE, a CYCLE LIMIT, and sometimes a KICKOFF ROUTINE. The STUDENT'S ROUTINE is the name of the routine the student is required to write to solve this problem. The CHECKING ROUTINE is a routine that checks to see if the student's program operated correctly. The CYCLE LIMIT is a limit imposed on the student's program. TIPL imposes this limit by setting W33 and supplying a trap routine for H3 on W26. The KICKOFF ROUTINE is a routine that is fired once at the very beginning of PHASE TWO. It is used, for example, in Problem 69 to modify J155 so that J155 checks each line the student prints. Most data sets do not have KICKOFF ROUTINES. If no CYCLE LIMIT is specified, then a STANDARD CYCLE LIMIT of 1000 cycles is imposed.

DATA SETS give a list of INPUTS to be placed in HO for the student's program to operate on. The STUDENT'S ROUTINE is then fired. When the STUDENT'S ROUTINE terminates, TIPL puts into HO an ANSWER, which is also supplied in the DATA SET. It then fires the CHECKING ROUTINE. The CHECKING ROUTINE sets H5 plus if the student's program was correct, minus if not. Another piece of information stored in a DATA SET is a SIGNAL. The SIGNAL specifies how the student's program was to have set H5--plus, minus, or don't care. Before TIPL puts any INPUTS into HO it counts available

space and after the CHECKING ROUTINE has finished (and presumably removed the ANSWER and the student's outputs from HO) it again counts available space. It then compares the difference to the integer data term DELTA SPACE that is stored in the DATA SET. TIPL then prints "DATA SET n CORRECT" or "DATA SET n INCORRECT". If H5 was set incorrectly, it also prints "H5 INCORRECT". If the difference in the available space counts did not equal DELTA SPACE, it also prints "AVAILABLE SPACE INCORRECT". TIPL also checks to see if the W's remained safe over the student's program; if not, it prints "W'S NOT KEPT SAFE".

TIPL then goes back to the PROBLEM STRUCTURE, finds the next DATA SET, and does the same thing for this DATA SET. After the last DATA SET it prints "PROBLEM n CORRECT" or "PROBLEM n INCORRECT" and "m CYCLES" where m is the number of cycles the student's program operated. This provides a rough estimate of how efficient the student's program was. It then again prints the card image of the student's Type-T card, which presumably has his name on it and hence provides identification at the end of his output. TIPL then goes on to the next student's program, or, if that is the end of the batch, it terminates.

If in any data set the student's program exceeds the CYCLE LIMIT, TIPL will print "TIPL IMPOSED CYCLE LIMIT

EXCEEDED", "PROBLEM n INCORRECT" and then it does J202. It then prints the Type-T card again and goes on to the next student's program.

An example of a typical TIPL run is given in Fig. 2.

PRINTER

The PROBLEM STRUCTURES may be printed by using Type-P cards. A Type-P card with no name will cause all the problems to be printed. A Type-P card with NAME = n will cause the PROBLEM DATA STRUCTURE for problem n to be printed. Any number of Type-P cards may be used. They must, of course, be preceeded by the KICKOFF DECK. An example of the output produced by a Type-P card is given in Fig. 3. The output is generated directly from the PROBLEM DATA STRUCTURES. If an old problem is modified or a new problem added it can be printed with a Type-P card. The output produced is helpful in debugging data sets. It is helpful for the instructor to know what TIPL is checking for and how these checks are being carried out. If TIPL is being used to grade students in a programming course, it is suggested that Type-P cards be kept confidential since their output gives excellent clues on how to cheat.

```
                              9
                              5          402
                              2 A                    2
                              2 B                    2
                              2 P                   33
                              2 T                    2
                              2 X                    7
ROBERT DUPCHAK PROB 12        T 12
                              5
PRINT LIST X6                   P32      10X6
                                           J151
LOCATE A1 ON LIST X6                     10X6
                                         10A1
                                           J62
STORE LOCATION IN WO                       J50
REPLACE A1 WITH B1                       10B1
                                         21WO
POP WO                                     J30
REMOVE T1 FROM LIST X6                   10X6
                                         10T1
                                           J69
PRINT LIST X6 AGAIN                      10X6
                                           J151  0
                              5            .0
X6     0
       S1
       A1
       T1
X6     0
       S1
       B1

            DATA SET 1 CORRECT

X6     0
       T1
       S1
       A1
X6     0
       S1
       B1

            DATA SET 2 CORRECT

PROBLEM 12 CORRECT      30 CYCLES

ROBERT DUPCHAK PROB 12   T 12
```

Fig. 2--Output Produced by the Deck Shown in Fig. 1

```
PROBLEM 11   R66 STUDENT'S ROUTINE   .3 CHECKING ROUTINE   1000 CYCLE LIMIT          11
DATA SET 1**************************************************************************
                    INPUT (0)
                                        S        0
                    INPUT (1)
                                5862*   9-0      0
                                                 S1
                                                 S2
              ---------------------------------------------------------------------
              5862* MUST BE THE SAME AS
                                5877*   9-0      0
                                                 S1
                                                 S2
                                                 S
               H2 MUST BE 1 CELL SHORTER   H5 MUST BE +
DATA SET 2**************************************************************************
                    INPUT (0)
                                        S        0
                    INPUT (1)
                                5889*   9-0      0
                                                 S2
                                                 S
                                                 S3
              ---------------------------------------------------------------------
              5889* MUST BE THE SAME AS
                                5903*   9-0      0
                                                 S2
                                                 S
                                                 S3
              H2 MUST BE THE SAME LENGTH   H5 MUST BE -
**************************************************************************************
```

Fig. 3--Printout for Problem 11

ADDING AND MODIFYING PROBLEM DATA STRUCTURES

Modification of TIPL or of the PROBLEM STRUCTURES or the addition of new problems can be accomplished by using Type-L and Type-S cards. A Type-L card causes TIPL to do a J165--LOAD ROUTINES AND DATA. When loading in this manner the final Type-5 card should kick off to period zero. A Type-S card with NAME = n will cause TIPL to do a J166--SAVE ON UNIT n FOR RESTART.

An example of a PROBLEM STRUCTURE is given in Fig. 4; its printout via a Type-P card was shown in Fig. 3.

The symbol in the head cell of the PROBLEM STRUCTURE names the PROBLEM PARAMETER sublist. This sublist must contain in its head the name of the CHECKING ROUTINE. If it is desirable to impose a CYCLE LIMIT other than the STANDARD CYCLE LIMIT, an integer data term should be added to the PROBLEM PARAMETER sublist. If no CYCLE LIMIT is desired, then the symbol .1 should be added. If a KICKOFF ROUTINE is to be imposed, then the CYCLE LIMIT must be given explicitly, followed by the name of the KICKOFF ROUTINE (.0 may be used to specify the STANDARD CYCLE LIMIT in this case). The PROBLEM PARAMETER sublist never has more then three symbols (including the symbol in its head cell).

```
                          5        01
PROBLEM PARAMETERS          .110     9-0
STUDENT'S ROUTINE                    R66
DATA SET ONE                         9-1
DATA SET TWO                         9-2     0
CHECKING ROUTINE          9-0        .3      0
DATA SET ONE              1
  INPUT (0)               9-1        SO
  INPUT (1)                          9-10
  ANSWER                             9-11
  DELTA SPACE                        9-19
  SIGNAL                             .4      0
                          9-10       0
                                     S1
                                     S2      0
                          9-11       9-10
                                     0
                                     S1
                                     S2
                                     SO      0
                          9-19  -01                1
DATA SET TWO              1
  INPUT (0)               9-2        SO
  INPUT (1)                          9-20
  ANSWER                             9-21
  DELTA SPACE                        9-29
  SIGNAL                             .3      0
                          9-20       0
                                     S2
                                     SO
                                     S3      0
                          9-21       9-20
                                     0
                                     S2
                                     SO
                                     S3      0
                          9-29  +01                0
```

Fig. 4--PROBLEM STRUCTURE for Problem 11

The first list cell on the main PROBLEM STRUCTURE list always contains the name of the STUDENT'S ROUTINE. If the student is not required to supply a routine but simply to supply a data structure, we would say the STUDENT'S ROUTINE is JO.

Following the name of the STUDENT'S ROUTINE on the main list of the PROBLEM STRUCTURE are the names of the DATA SETS.

Each DATA SET is given as a list. The list begins with any number of INPUTS (including zero) for the student's program. Immediately following the INPUTS comes the ANSWER. The ANSWER can be a symbol, the name of a list, or the name of a structure. Following the ANSWER is DELTA SPACE. DELTA SPACE is either an integer data term or the symbol .0, meaning "don't check available space." The next element on the DATA SET list is either .2 or .3 or .4. This is the SIGNAL. .3 means the student's program must leave H5 minus; .4 means the student's program must leave H5 plus; .2 means it doesn't matter how the student's program leaves H5. Instead of using .2, this last symbol can be deleted from the DATA SET list and the effect will be the same.

Another example of a PROBLEM STRUCTURE is given in

Fig. 5 and its printout is given in Fig. 6. This example shows one additional convention (our last). Following the SIGNAL there may be any even number of symbols on the DATA SET list. Before the student's program is executed, TIPL reads these symbols in pairs and sets the first symbol of each pair identical to the second symbol in the pair (using J121). The necessity for this ability is obvious in this example. It should be noted that this feature can be used not only for supplying data structures with particular names but also for supplying routines with given names. Thus, in Problem 37, routine P88 is supplied to the student as routine .371. P88 cannot be loaded directly since in another problem the student may use cell P88 for another purpose and indeed in Problem 35 he is required to code routine P88.

The following TIPL routines and cells may be found useful when adding new problems.

CHECKING ROUTINE .1--This routine normally has two inputs, (0) is normally the ANSWER and (1) is normally the student's output. The routine tests if structure (0) is identical to structure (1). Symbols must be of the same type and data terms must be equal. Non-local symbols must be identical. If (0)

```
                          5          01
PROBLEM PARAMETERS          .120       9-0
STUDENT'S ROUTINE                      P32
DATA SET ONE                           9-1
DATA SET TWO                           9-2     0
CHECKING ROUTINE            9-0        .3      0
DATA SET ONE              1
  ANSWER                    9-1        9-10
  DELTA SPACE                          9-99
  SIGNAL                               .2
  MAKE THIS IDENTICAL                  X6
    TO THIS                            9-11    0

                            9-11       0
                                       S1
                                       A1
                                       T1      0
                            9-10       X6
                                       0
                                       S1
                                       B1      0

DATA SET TWO              1
  ANSWER                    9-2        9-20
  DELTA SPACE                          9-99
  SIGNAL                               .2
  MAKE THIS IDENTICAL                  X6
    TO THIS                            9-21    0

                            9-21       0
                                       T1
                                       S1
                                       A1      0
                            9-20       X6
                                       0
                                       S1
                                       B1      0
                            9-99 +01                   1
```

Fig. 5--PROBLEM STRUCTURE for Problem 12

```
PROBLEM 12    P32 STUDENT'S ROUTINE    .3 CHECKING ROUTINE    1000 CYCLE LIMIT          12
DATA SET 1**********************************************************************************
                    SUPPLIED
                                        X6        0
                                                  S1
                                                  A1
                                                  T1
                  ------------------------------------------------------------------------
                  X6 MUST BE THE SAME AS
                                  5944*  9-0       0
                                                   S1
                                                   B1
                  H2 MUST BE 1 CELL LONGER
DATA SET 2**********************************************************************************
                    SUPPLIED
                                        X6        0
                                                  T1
                                                  S1
                                                  A1
                  ------------------------------------------------------------------------
                  X6 MUST BE THE SAME AS
                                  5967*  9-0       0
                                                   S1
                                                   B1
                  H2 MUST BE 1 CELL LONGER
**********************************************************************************
```

Fig. 6--Printout for Problem 12

is J8, this routine removes J8 from H0 and sets H5 plus. This feature is useful in Problem 29 for example.

CHECKING ROUTINE .2--This routine normally has two inputs, (0) is normally the ANSWER and (1) is normally the student's output. The routine tests if list (0) and list (1) contain the same symbols, independent of order. This routine observes the J8 convention of .1.

CHECKING ROUTINE .3--This routine has one input, (0), which is normally the ANSWER. It assumes that (0) is a cell whose SYMB names a structure and whose LINK names another structure. It tests if these two structures are identical by using .1. It also observes the J8 convention.

CHECKING ROUTING .4--This routine has one input, (0), which is normally the ANSWER. It assumes that (0) is a cell whose SYMB names a list and whose LINK names another list. It tests if these two lists are the same, independent of order, by using .2. It also observes the J8 convention.

DATA SET FLAG .5-----At the beginning of each data set a J4 is stored into cell .5. If at anytime this is replaced by a J3, the data set will be marked incorrect.

This is used, for example, in the problems on generators. TIPL supplies a subprocess to the student's generator which, among other things, tests if the W's are in proper context. If they are not, it signals an error by putting J3 in .5.

HO DEPTH TESTER .8---This routine assumes (0) is an integer data term n. After removing this input it tests if HO is n cells deep. If so, it sets H5 plus. If not, it removes everything from HO and sets H5 minus. This routine is used at the beginning of CHECKING ROUTINES to make sure the student left enough outputs. This prevents the CHECKING ROUTINE from clobbering out because of too few inputs.

CYCLE STOPPER .9-----This routine is identical to J1 except that H3 is increased by only one cycle no matter how many cycles routine (0) takes. All routines that TIPL supplies to the student are executed by .9 and thus do their work in one cycle. This makes the cycle count given at the end of the student's output a truer reflection of the efficiency of his routine.

ASSIGNMENTS

When printing out a problem that has a KICKOFF ROUTINE .n it is desirable to also print a comment describing what routine .n does. This is done by making a list of print lists (like L17, Problem 69) with the regional name .m. This structure is loaded (with a Type-L card) and .m is ASSIGNED to .n with a Type-A card, NAME = .n, SYMB = .m. An example of this feature can be seen in Problem 69, where .694 is ASSIGNED to .691. This feature is optional--KICKOFF ROUTINES are not required to have assignments.

Likewise, CHECKING ROUTINES can have assignments. However, routines, not lists, are ASSIGNED to CHECKING ROUTINES. The printer, if it finds an assignment for a CHECKING ROUTINE, will place in (0) the ANSWER and then it will fire the ASSIGNED routine. An example can be seen in Problem 22 where .222 is ASSIGNED to CHECKING ROUTINE .221. If the printer does not find an assignment for CHECKING ROUTINE .n it will print ".n IS APPLIED TO" and then it will print the ANSWER with J150.

POSSIBLE MODIFICATIONS REQUIRED BY LIMITED SYSTEMS

The code for .8 does a J126 on HO. This may not work on your machine, or, if it does, it may not work in the

same manner. Therefore .8 may have to be recoded. If this is difficult, an easy way out is to code it .8 J8 J4. This may cause the checking routines to clobber out when the student does not leave enough outputs. This may confuse the student, but it is the only ill effect.

TIPL detects undefined symbols by modifying region control words, using routines J175 and J197. This is all done in subroutine 9-22 in routine .0. 9-22 has two inputs. (0) is the zeroth element of a region and (1) is an internal symbol. 9-22 sets the maximum size of region (0) to (1), where (1) is interpreted as an integer. If your system does not have J175 and/or J197 this subroutine will have to be recoded. If this is difficult to do, the easy way out is to code it 9-22 J8 J8. The only ill effect here is that any error the student makes in his Type-2 cards will go undetected. This is the one and only place in the entire program that a J19n or a block manipulation primitive is used.

TIPL assumes five characters are packed per BCD data term. If your System packs four characters, or more than five characters, per BCD data term, then the PROBLEM STRUCTURES for Problems 73-75 will have to be modified. If your System packs fewer than four characters, you are

in great trouble. All the print lists in .0 will have to be recoded and many of the PROBLEM STRUCTURES will have to be modified.

TIPL assumes a 120-character print line. TIPL will function properly with any other size print line of 72 or more characters, except that Problem 71 will have to be modified and some information will be lost when making Type-P card problem printouts.

TIPL assumes one character is packed per word in a print line. If this is not so on your machine, the LINK fields of the Type-3 cards reserving print lines .6 and .7 will have to be modified.

TIPL uses about 12,100 words of core: 4500 words are reserved by Type-2 and Type-3 cards; routines .0 through .9 use 1600 words; and the data sets use about 6000 words. This storage requirement may be reduced in two ways if necessary: (1) the Type-2 and Type-3 cards can reserve fewer words; and (2) most of the data sets can be put in auxiliary storage.

1000 words can be gained by not reserving the special character regions (the period region must of course be reserved). The student should then of course be instructed that he can only use alphabetic regions. Another 240 words

can be gained by reserving the period region only up to 760. More space can be gained by reducing the size of the alphabetic regions. Note however that the zero element of every region (special character regions included) must be defined and that (with the exception of Problem 71) A8, B5, C1, D99, L100, N10, P97, Q23, R87, S8, T13, V2, X12, Y11, and Z7 are the highest subscript elements used in each of these regions. Problem 71 uses the 100th element of each of the alphabetic regions and hence it would have to be modified if the size of the alphabetic regions is reduced.

The second method of reducing the amount of core required is to put most of the data sets on auxiliary storage. Not all the data sets can be put on auxiliary since some of them are not true IPL data structures. The data structures that can be put on auxiliary are listed in Fig. 7. To do this, simply insert Type-7 cards with Q = 5 in front of each of these data structures. Doing this will reduce the core requirement by 3850 words. The executive routine .0 will call in these data structures via J105 and thus the cost in running time of this modification will be one (and only one) auxiliary storage reference per problem.

An additional 1875 words of core can be saved by loading the routines listed in Fig. 7 into auxiliary. This modification can be made by reserving a 240-word buffer for auxiliary routines (and thus reducing the profit of this modification to 1635 words) and inserting Type-7 cards with Q = 4 in front of each of these routines. The cost of this modification in running time will also be one (and only one) auxiliary storage reference per problem.

DATA STRUCTURES THAT CAN BE PUT IN AUXILIARY:		ROUTINES THAT CAN BE PUT IN AUXILIARY:
.110	.450	.221
.120	.460	.222
.130	.470	.301
.140	.480	.302
.150	.490	.441
.160	.500	.442
.170	.510	.461
.180	.520	.462
.190	.530	.481
.200	.540	.521
.210	.550	.522
.220	.560	.531
.230	.570	.532
.240	.580	.592
.250	.600	.593
.260	.610	.603
.270	.620	.613
.280	.630	.633
.300	.640	.643
.310	.647	.691
.320	.670	.697
.330	.680	.721
.340	.690	.751
.360	.694	.752
.370	.700	
.380	.710	
.390	.720	
.400	.724	
.403	.730	
.410	.740	
.420	.750	
.440		

Fig. 7--Data and Routines that can be Put in Auxiliary

THE TIPL DECK

The TIPL deck is sequenced in cols. 77-80. The first card is 0001 and the last card is 8849. It must be preceded by a KICKOFF DECK, as described at the beginning of this section. Card 7970 is a final Type-5 KICKOFF card. Cards 7971-7988 are Type-A cards. Card 7989 is a Type-S card with NAME = 2. It causes a SAVE ON UNIT 2 FOR RESTART. Card 7990 is a Type-P card with blank NAME. This causes the data sets for all the problems to be printed out.

Cards 7991-8849 consist of two series of tests. The first series consists of 11 complete "solutions" to Problem 11. Each of these "solutions," except the last one, contains an error. The error each of these "solutions" contains is gang-punched in cols. 6-40. TIPL, if it is functioning correctly on your machine, should detect each of these errors and take appropriate action.

The second series of tests consists of correct solutions to 27 problems. These 27 problems use every routine in TIPL. These solutions should not be used as examples for the student since some of the solutions cheat (use TIPL routines to solve problems) and others, especially the early ones, use J's that have not yet been introduced to the student. TIPL, if it is functioning correctly on your machine, should mark each of these solutions correct.