DISCLAIMER

Although each program has been tested by its contributor, no warranty, express or implied, is made by the contributor or 1620 USERS Group, as to the accuracy and functioning of the program and related program material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the contributor or 1620 USERS Group, in connection therewith.

1620 USERS GROUP PROGRAM REVIEW AND EVALUATION

(fill out in typewriter or pencil, do not use ink)

Program No._____ Date_____

Program Name:_____

1. Does the abstract adequately describe what the program is and what it does? Yes___ No___
   Comment_____

2. Does the program do what the abstract says? Yes___ No___
   Comment_____

3. Is the Description clear, understandable, and adequate? Yes___ No___
   Comment_____

4. Are the Operating Instructions understandable and in sufficient detail? Yes___ No___
   Comment_____
   Are the Sense Switch options adequately described (if applicable)? Yes___ No___
   Are the mnemonic labels identified or sufficiently understandable? Yes___ No___
   Comment_____

5. Does the source program compile satisfactorily (if applicable)? Yes___ No___
   Comment_____

6. Does the object program run satisfactorily? Yes___ No___
   Comment_____

7. Number of test cases run_____. Are any restrictions as to data, size, range, etc. covered adequately in description? Yes___ No___
   Comment_____

8. Does the Program Meet the minimal standards of the 1620 Users Group? Yes___ No___
   Comment_____

9. Were all necessary parts of the program received? Yes___ No___
   Comment_____

10. Please list on the back any suggestions to improve the usefulness of the program. These will be passed onto the author for his consideration.

Please return to:

Mr. Richard L. Pratt
Data Corporation
7500 Old Xenia Pike
Dayton, Ohio 45432

Your Name _____

Company _____

Address _____

User Group Code_____

THIS REVIEW FORM IS PART OF THE 1620 USER GROUP ORGANIZATION'S PROGRAM REVIEW AND EVALUATION PROCEDURE. NONMEMBERS ARE CORDIALLY INVITED TO PARTICIPATE IN THIS EVALUATION.

11/09/64

A3

ABSTRACT

LINEAR PROGRAMMING I

May 15, 1962

by

F. W. Wood

Modifications or revisions to this program, as they occur, will be announced in the appropriate Catalog of Programs for IBM Data Processing Systems. When such an announcement occurs, users should order a complete new program from the Program Information Department.

NATIONAL STEEL CORPORATION
Research and Development Department
Weirton, West Virginia

ABSTRACT

LINEAR PROGRAMMING I

by

F. W. Wood

DESCRIPTION

This program minimizes, or maximizes by negating the functional coefficients, a linear functional subject to a set of linear restrictions.

Maximum problem size is determined by the formula $(m + 1)(n + 2) = 4464$ where m is the number of restrictions plus the cost row and n is the number of variables, including slacks needed, plus the right hand side. The 4464 is based on FORTRAN with a lowered origin.

Shadow cost type out and final matrix punchout are under control of program switches. All output is via the typewriter with the exception of the final matrix punchout.

The calculations utilize floating point arithmetic with an 8-digit mantissa. Zero values are not used in computation. The method of the artificial basis is used to find the first feasible solution and the standard Simplex algorithm is used until optimality is attained.

Program input is punched in standard 1620 FORTRAN format.

MINIMUM COMPUTER

Any computer with the ability to compile 1620 FORTRAN with FORMAT. The maximum size inequality depends on the configuration of the computer in question. The above formula was derived for a machine with automatic divide hardware and 60 K memory.

PROGRAM LANGUAGE

FORTRAN

RUNNING TIME

Varies with the density of the matrix and the number and type of iterations made.

DESCRIPTION OF PROGRAM

PURPOSE

The main purpose of this program is to determine non-negative values of a number of variables which give a minimum (maximum) value to A_o where A_o is defined by a relationship of the form

$$A_o = A_1 X_1 + A_2 X_2 + \text{--------} A_n X_n$$

where the A_j's are known coefficients and the X_j's are the unknown variables subject to a set of simultaneous equations which represent the conditions of the problem where the matrix of the coefficients and the requirement vectors are known quantities.

PROGRAM COMPILATION

It will be necessary to compile the source statements for each different machine configuration and also for the approximate problem size in question. The reason the approximate problem size is mentioned is that any problem up to the compiled size may be run and only that part of the reserved memory area will be used.

The first source statement of each phase contains a Dimension statement with A(I,J), W(I), and L(I). To change problem size, these lists must be changed and the program recompiled. The subscripts in the above lists are defined as follows:

I = number of equations + 2 (the functional is excluded as an equation.)

J = number of variables, including slacks, + 1

All phases, if they are to be used, must be compiled with the same dimension statements.

INPUT FORMAT

All input to the program must conform to the specificiations for fixed and floating point constants as outlined in the 1620 FORTRAN WITH FORMAT MANUAL. Input and Output information varies with the Phase of the program being utilized at the time, so it will be spelled out in the detailed Phase descriptions below.

DATA LOADER PHASE

INPUT

The input to the Data Loader Phase is as follows:

Header Card:

Card 1. cc

1-4 The number of equations + 1 (excluding the cost row.)

5-8 The number of variables (including slacks) + 1

Matrix Elements: (One element per card)

cd.2 cd n + 1 n is the number of non-zero elements.

cc

1-4 Row number of element in the matrix.

5-8 Column number of the element in the matrix.

9-28 The value of the element in the matrix (a decimal point must be punched somewhere in the field.)

Right Hand Side: (One element per card)

cd n + 3 - cd. n + 3 + (II - 1)

cc

1-4 Row number of the equation

5-8 1) 0000 if the equation for that row is a strict equality; or

2) If the equation is an inequality, put the column number of slack associated with the row in this field.

9-28 Value of the right hand side element. *A decimal point must be punched somewhere in the field.)

* Only positive right hand side elements are allowed.

It should be noted that card n + 2 is missing in the above input description. This is to be a zero (or blank) card that is to be inserted between the last matrix element and the first right hand side card. Another zero (or blank) card is to follow the right hand side elements.

The costs, in the functional, are to be entered in row 0001 and the column corresponding to the variable in question. The first restricting equation is then row 0002 of the matrix.

Slack vectors must be entered as separate variables. If it is desired to put a cost on a slack vector, this is possible by entering the cost into row 0001 and the appropriate column.

Zero matrix elements do not have to be entered and the rest of the elements can be entered in any order.

OUTPUT

The only output of this Phase is the typing of LOAD NEXT PHASE to specify that the computer is ready for the next Phase.

OPERATING INSTRUCTIONS

1) Clear memory.

2) Load object deck as any other FORTRAN deck.

3) Load Data in following order:

   a) Header Card

   b) Matrix elements (including costs)

   c) Zero Card

   d) Right Hand Side elements

   e) Zero Card

When the data is loaded, LOAD NEXT PHASE will be typed and the computer will halt. The next phase can now be loaded.

SOLUTION PHASE

Description

The method of the artificial bases is used until a first feasible solution is reached, if all restrictions are not inequalities. If the problem is not

feasible, the typewriter will type INFEASIBLE and the computer will come to a halt. Depression of the Start Key will cause the solution up to that point to be typed and the other output will be typed or punched, depending upon the switch settings.

If the problem has a feasible solution after a number of Type 1 iterations, the word FEASIBLE will be typed and the iteration counter will be reset. At this time the Simplex Algorithm takes control until the optimal solution is reached or the problem is determined to be UNBOUNDED. If the latter is the case, and it is desired to find out which column is all negative, it is necessary to type out the contents of KJ as defined in the Symbol Table output of the FORTRAN compilation.

Variables are identified by the column number associated with them.

The solution of a problem can be interrupted at any time by the use of Program Switch 4. When Program Switch 4 is turned on, the program will complete the iteration it is on and then proceed to type the solution at that time, type the shadow costs at that time if Program Switch 3 is on, and punch the matrix at that time if Program Switch 2 is on.

If it is desired to stop a run in the middle and to complete it at some later time, turn Switch 2 and Switch 4 on. This will cause the Simplex Algorithm to terminate and initiate the matrix punchout. This punchout alone with the header card can then be loaded back with the Loader Phase. The Solution Phase can then be loaded and solution of the problem can continue to the optimal solution. Program Switch 1 must be turned on when the Solution Phase is loaded to continue a run in the manner mentioned above.

Matrix punchout, after the optimal solution has been reached, can be used directly with a header card to affect cost changes or right hand side changes as described under these sections.

Input

There is no input to the Solution Phase. It operates on data already in memory that was either loaded by the Loader Phase or transformed by the Cost Changer Phase.

Output

During the running of the program, the iteration number, value of the functional, and the variable introduced into the basis on that iteration, is typed in the following manner:

ITER.	FUNCTIONAL	VAR. IN	(heading)
1	200.00	2	

When the optimal solution is reached, the following is typed out:

FUNCTIONAL	200.00000000
VARIABLE	VALUE
1	10.00000000
2	20.00000000

If the shadow costs are called for, the following is typed. Zero elements are not typed:

VARIABLE	SHAD. COST
3	1.00000000
4	1.00000000

The matrix elements are punched as follows:

cc 1-4 row number cc 5-8 column number cc 9-28 value of the element

Operating Instructions

This Solution Phase is to be used after the Loader Phase or the Cost Changer Phase; therefore, do not clear memory.

1) Set Switches.

	ON	OFF
Switch 1	For restart problems	For initial start problems
Switch 2	To punch final matrix	To ignore final matrix punch
Switch 3	Type out shadow costs	Ignore shadow cost type out
Switch 4	To interrupt program and punch restart matrix	To run problem to optimal solution

2) Push Reset on 1620 console.

3) Load Program in 1622 read hopper.

4) Push Load on 1622 read-punch.

5) Push Start on 1620 console when LOAD DATA is typed on the console typewriter.

COST CHANGER PHASE

Description

Once the optimal solution has been reached, it is possible to change the costs of the variables whether they are or are not in the final solution. The outcome of such a change has two possible consequences:

1) The solution is still optimal and only the functional changes if the variable is in final solution; or

2) The solution is no longer optimal.

If situation (1) above exists, the new functional is typed and the computer halts. Depression of the Start Key on the 1620 console will transfer control back to read statement to read the next cost change. This allows one to parameterize a certain cost by incrementing the input values each time until situation (2) above is experienced. When situation (2) is reached, LOAD SOLUTION DECK is typed and the computer halts. At this point, more cost changes can be made by depressing Start or the Solution Phase can be loaded to perform the necessary iterations.

The cost changes can be made directly after the Solution Phase or after the final matrix punchout has been loaded with the Data Loader Phase.

Input

The input to the cost changer consists of one type of cost changer card. The format of this card is:

cc 1-4	Variable whose cost is to be changed.
cc 5-14	Old cost with a decimal point somewhere in the 10-digit field.
cc 15-24	New cost with a decimal point somewhere in the 10-digit field.

A blank card must follow the last cost change card if more than one variable is involved in the change. If parameterization is desired, each cost change must be followed by a blank card.

If a is the step desired for parameterizing variable 0001, the cards should be prepared as follows:

	cc 1-4	cc 5-14	cc 15-24
Card 1	0001	old cost	old cost + a
Card 2	blank		
Card 3	0001	old cost + a	old cost + 2a
Card 4	blank		

and so on until the range in question is covered.

If several cost changes are to be carried out at the same time, the cost change cards associated with these variables should be put together and a single blank card should follow them.

Output

The output from this phase includes the variable whose cost is to be changed, its old cost, its new cost, if the solution is still optimal. STILL OPTIMAL is typed and also the new functional, and if more iterations are necessary, LOAD SOLUTION DECK is typed.

*If the original costs were negated to minimize instead of maximize, it must be remembered that the old cost to be entered is a negative cost and the same holds for the new cost if the above applies.

Operating Instructions

As stated above, this phase can be used directly after the Solution Phase or with the final matrix punchout. If the Cost Changer Phase is to follow the Solution Phase, the following instructions should be carried out:

1) Do not clear memory.
2) Load deck in 1622 read hopper.
3) Load cost change cards.
4) Place blank card after cost change cards.
5) Press load button on 1622.
6) Push Start Key on 1620 console when LOAD DATA is typed.

If the final matrix is used, load the matrix as explained under the Data Loader Phase and then proceed with the above steps.

If the solution deck is needed, load it in the 1622 read hopper and push the Load Key. Switch settings should be as explained under Solution Phase.

RIGHT HAND SIDE CHANGER PHASE

Description

This program phase assumes optimality of the primal problem and should be used when changes to the right hand side vector are made or when all costs are positive and at least one right hand side element is negative.

The dual algorithm is utilized for the problem solution.

If changes in the right hand side are to be made, this phase should follow the Solution Phase, the Cost Changer Phase if no iterations are required, or the Data Loader Phase if the final matrix punchout is used.

It is necessary that, at the initial solution of the problem, a positive unit vector, either slack or artificial, is present in the tableau that corresponds to the row in which the change is to be made. This vector is used in computing a new right hand side.

If an artificial vector is used to make the right hand side change, the value of the resulting functional will not be correct and must be computed by hand. A correct functional is formed if slack vectors are used in the change.

An artificial vector must be entered with an arbitrary high positive cost to insure that they will not enter the basis. If artificials are used, the method to find the first feasible solution can bring these artificials into the basis and will type FEASIBLE before they are removed. Therefore, feasibility is determined by the absence of such artificials in the final optimal solution. If an artificial is still present, the solution is not feasible.

See Problem No. 2 for an example of entering artificials.

If the initial problem to be solved contains all positive cost elements and negative right hand side elements, the data should be loaded with the Data Loader Phase and then the Right Hand Side Changer Phase should be loaded and followed by a single blank card that signals the program to start the algorithm. In this manner, the dual algorithm can be used for problem solution if the initial conditions warrant it.

The program solves the problem and comes to a halt. Depression of the Start Key on 1620 console causes control to be transferred to the read statement that reads the next change card. Parameterization of a right hand side element can take place in the same manner as explained under the Cost Changer Phase.

Input

The input to this phase consists of a single type of card:

cc	
1-4	1) 0000 if the above unit vector is artificial. 2) 0001 if the above unit vector is slack.
5-8	Column number that contained a unit vector at start of problem.
9-18	Old right hand side element with a decimal point somewhere in the 10-digit field.

19-28	New right hand side element with a decimal point somewhere in the 10-digit field.

As many of these cards can follow one another as desired.

Output

The output from this phase consists of two types. The first type is the old and the new right hand side element type out for verfication. The second type is similar to the output of the Solution Phase.

Operating Instructions

Do Not clear memory.

1) Set switches. (same as for Solution Phase)

2) Load object deck in 1622 read hopper.

3) Load change cards followed by a blank card.

4) Push load on 1622.

EXAMPLES OF EQUATION SETUP

1) $X_1 + X_2 + X_3 \leq 10$

Add positive slack vector X_4 to the matrix and set up the right hand side element card as follows:

cc	
1-4	Row number
5-8	0004
9-28	10.

2) $X_1 + X_2 + X_3 = 10$

No additions are required to the matrix. The right hand side card is then:

cc	
1-4	Row number
5-8	0000
9-28	10.0

3) $X_1 + X_2 + X_3 \geq 10$.

Add negative slack vector to the matrix and set up the right hand side

element card as follows:

cc	
1-4	Row number
5-8	0000
9-28	10.

It should be noted that this negative slack cannot be used for right hand side changes.

This program is also available in a 5-phase setup that allows about 3,000 more digits of matrix storage for larger problems. Input, output, and general format is the same as the above program.

Sample Problem No. 1

Maximize $2X_1 + 4X_2 + X_3 + X_4$

Subject to the following conditions:

$$X_1 + 3X_2 + X_4 \leq 4$$

$$2X_1 + X_2 \leq 3$$

$$X_2 + 4X_3 + X_4 \leq 3$$

$$X_j \geq 0$$

Transform this to the following by negating the costs and adding slack vectors:

Minimize $-2X_1 - 4X_2 - X_3 - X_4$

Subject to the following conditions:

$$X_1 + 3X_2 + X_4 + X_5 = 4$$

$$2X_1 + X_2 + X_6 = 3$$

$$X_2 + 4X_3 + X_4 + X_7 = 3$$

$$X_j \geq 0$$

The tableau to be entered is:

Row	Column 1	2	3	4	5	6	7	
1	-2	-4	-1	-1				
2	1	3		1	1			4
3	2	1				1		3
4		1	4	1			1	3

INPUT SAMPLE PROBLEM NO. 1

```
00040008
00010001-2.
00010002-4.
00010003-1.
00010004-1.
000200011.
000200023.
000200041.
000200051.
000300012.
000300021.
000300061.
000400021.
000400034.
000400041.
000400071.
000000          ZERO OR BLANK CARD
000200054.
000300063.
000400073.
000000          ZERO OR BLANK CARD
```

OUTPUT SAMPLE PROBLEM NO. 1

```
LOAD DATA
DATA LOADED
LOAD DATA
ITER.      FUNCTIONAL          VAR. IN
FEASABLE
   1          5.33               2
   2          5.74               3
   3          6.49               1
FUNCTIONAL          6.49999980
VARIABLE            VALUE
   2          1.00000000
   1          1.00000000
   3           .50000000
VARIABLE      SHAD. COST
   4           .34999997
   5          1.10000000
   6           .45000000
   7           .25000000
```


Sample Problem No. 2

Minimize $X_1 + 6X_2 - 7X_3 + X_4 + 5X_2$

Subject to the following conditions:

$$5X_1 - 4X_2 + 13X_3 - 2X_4 + X_5 = 20$$

$$X_1 - X_2 + 5X_3 - X_4 + X_5 = 8$$

$$X_j \geq 0$$

The tableau is:

		Column 1	2	3	4	5	
	1	1	6	-7	1	5	
Row	2	5	-4	13	-2	1	20
	3	1	-1	5	-1	1	8

If the right hand side is to be changed, the following tableau must be entered:

		1	2	3	4	5	6	7	
	1	1	6	-7	1	5	∞	∞	
Row	2	5	-4	13	-2	1	1	0	20
	3	1	-1	5	-1	1		1	8

An arbitrary high cost must be entered for vectors 6 and 7 so that they will not enter the basis.

INPUT SAMPLE PROBLEM NO. 2

```
00030006
000100011.
000100026.
00010003-7.
000100041.
000100055.
000200015.
00020002 4.
0002000313.
00020004-2.
000200051.
000300011.
00030002 1.
000300035.
00030004 1.
000300051.
0000                          ZERO OR BLANK CARD
0002000020.
000300008.
0000                          ZERO OR BLANK CARD
```


OUTPUT SAMPLE PROBLEM NO. 2

```
LOAD DATA
DATA LOADED


LOAD DATA
ITER.      FUNCTIONAL          VAR. IN
   1            -.30                3
   2            -.00                5
FEASABLE
   1            8.57                2
FUNCTIONAL          8.57142730
VARIABLE          VALUE
   3          1.71428580
   2           .57142896
VARIABLE     SHAD. COST
   1         10.28571500
   4          1.57142920
   5          1.14285600
```

Sample Problem No. 3

Change the cost of X_2 in Sample Problem No. 1 from minus 4 to plus 1 using the final matrix punchout of Problem No. 1

Sample Problem No. 4

Change the requirement of the first equation in Sample Problem No. 1 from 4 to 20 using the final matrix punchout from Problem No. 1.

F W Wood
Mathematician

FWW/kh
5/15/62

INPUT SAMPLE PROBLEM NO. 3

FINAL MATRIX PUNCHOUT FROM SAMPLE PROBLEM NO. 1

```
00040008                                  HEADER CARD TO BE ADDED
   3   1          1.00000000
   2   2          1.00000000
   4   3          1.00000000
   1   4           .34999997
   2   4           .39999999
   3   4          -.19999999
   4   4           .15000001
   1   5          1.10000000
   2   5           .39999999
   3   5          -.19999999
   4   5          -.09999999
   1   6           .45000000
   2   6          -.19999999
   3   6           .59999998
   4   6           .04999999
   1   7           .25000000
   4   7           .25000000
                              BLANK CARD SUPPLIED BY PROGRAM
   1   0          6.49999980
   2   2          1.00000000
   3   1          1.00000000
   4   3           .50000000
   5   0           .00000000
                              BLANK CARD TO BE ADDED
```

INTPUT TO COST CHANGER

```
0002-4.      1.
0000                          ZERO OR BLANK CARD
```

OUTPUT SAMPLE PROBLEM NO. 3

```
LOAD DATA
DATA LOADED


LOAD DATA
VAR. OLD COST   NEW COST
   2  -4.00000   1.00000
LOAD SOLUTION DECK


LOAD DATA
ITER.     FUNCTIONAL        VAR. IN
FEASABLE
   1          5.62              4
FUNCTIONAL          5.62500010
VARIABLE          VALUE
   4          2.50000000
   1          1.49999990
   3           .12499997
VARIABLE     SHAD. COST
   2          4.12500030
   5           .75000010
   6           .62499990
   7           .25000000
```


INPUT SAMPLE PROBLEM NO. 4

FINAL MATRIX PUNCHOUT FROM SAMPLE PROBLEM NO. 1

```
00040008                          HEADER CARD TO BE ADDED
   3   1          1.00000000
   2   2          1.00000000
   4   3          1.00000000
   1   4           .34999997
   2   4           .39999999
   3   4           .19999999
   4   4           .15000001
   1   5          1.10000000
   2   5           .39999999
   3   5           .19999999
   4   5           .09999999
   1   6           .45000000
   2   6           .19999999
   3   6           .59999998
   4   6           .04999999
   1   7           .25000000
   4   7           .25000000
                                  BLANK CARD SUPPLIED BY PROGRAM
   1   0          6.49999980
   2   2          1.00000000
   3   1          1.00000000
   4   3           .50000000
   5   0           .00000000
                                  BLANK CARD TO BE ADDED
```

INPUT TO THE RIGHT HAND SIDE CHANGER

```
000100054.          20.
0000                          ZERO OR BLANK CARD
```

OUTPUT SAMPLE PROBLEM NO. 4

```
LOAD DATA
DATA LOADED


LOAD DATA
      OLD RHS              NEW RHS
              4.00000                20.00000
ITER       FUNCTIONAL       VAR. IN
   1          20.25              4
   2          12.00              5
FUNCTIONAL          12.00000000
VARIABLE       VALUE
   2          3.00000030
   4           .00000000
   5         10.99999900
VARIABLE     SHAD. COST
   1          4.00000040
   3          3.00000040
   6          3.00000010
   7          1.00000010
```


DATA LOADER PHASE

```
08000       DIMENSION A(44,96),W(44),L(44),II(1),JJ(1),III(1),X(1)
08000       DIMENSION JK(1),KJ(1),KKK(1),IJ(1),K(1),I(1),J(1),XMIN(1)
08000     1 FORMAT(I4,I4)
08028     2 FORMAT(I4,I4,F20.8)
08060     4 FORMAT(11HDATA LOADED)
08106   108 READ 1,II,JJ
08142       III=II+1
08178       DO 10 I=1,III
08190       W(I)=0.0
08238       L(I)=0
08286       DO 10 J=1,JJ
08298    10 A(I,J)=0.0
08454    11 READ 2,I,J,X
08502       IF(I)16,16,18
08558    18 A(I,J)=X
08642       GO TO 11
08650    16 J=JJ
08674    17 READ 2,I,JK,X
08722       IF(I)20,20,19
08778    19 A(I,JJ)=X
08862       L(I)=JK
08910       GO TO 17
08918    20 PRINT 4
08942       PAUSE
08954       END
```

COST CHANGER PHASE

```
08000       DIMENSION A(44,96),W(44),L(44),II(1),JJ(1),III(1),X(1)
08000       DIMENSION JK(1),KJ(1),KKK(1),IJ(1),K(1),I(1),J(1),XMIN(1)
08000     1 FORMAT(I4,F10.5,F10.5)
08032     2 FORMAT(13HSTILL OPTIMAL)
08082     3 FORMAT(10HFUNCTIONAL,F20.8)
08132     4 FORMAT(18HLOAD SOLUTION DECK)
08192   100 FORMAT(25HVAR. OLD COST   NEW COST )
08266     9 PRINT 100
08290    10 READ 1,JK,W(1),W(III)
08362       L(1)=JK
08386       IF(JK)11,50,11
08442    11 PRINT 1,JK,W(1),W(III)
08514       W(1)=W(III)-W(1)
08574       A(1,JK)=A(1,JK)+W(1)
08706       DO 15 I=2,II
08718       IF(L(I)-L(1))15,16,15
08810    15 CONTINUE
08846       GO TO 10
08854    16 DO 20 J=1,JJ
08866       IF(A(I,J))19,20,19
08982    19 A(1,J)=A(1,J)-W(1)*A(I,J)
09198    20 CONTINUE
09234       GO TO 10
09242    50 IJ=JJ-1
09278       DO 55 J=1,IJ
09290       IF(A(1,J))60,55,55
09394    55 CONTINUE
09430       PRINT 2
09454       PRINT 3,A(1,JJ)
09526       PAUSE
09538       GO TO 9
09546    60 PRINT 4
09570       PAUSE
09582       GO TO 9
09590       END
```


SOLUTION PHASE

```
08000       DIMENSION A(44,96),W(44),L(44),II(1),JJ(1),III(1),X(1)
08000       DIMENSION JK(1),KJ(1),KKK(1),IJ(1),K(1),I(1),J(1),XMIN(1)
08000     2 FORMAT(I4,I4,F20.8)
08032     5 FORMAT(I4,F20.8)
08060     6 FORMAT(10HINFEASABLE)
08104     7 FORMAT(23HVARIABLE          VALUE)
08174     8 FORMAT(10HFUNCTIONAL,F20.8)
08224     9 FORMAT(23HVARIABLE     SHAD. COST)
08294   100 FORMAT(1H )
08320   101 FORMAT(35HITER.     FUNCTIONAL        VAR. IN)
08414   105 FORMAT(I4,F15.2,10X,I4)
08474       PRINT 101
08498       KKK=0
08522       IF(SENSE SWITCH 1)40,22
08542    22 I=1
08566    23 I=I+1
08602       IF(I-III)24,40,40
08670    24 IF(L(I))23,25,23
08750    25 DO 27 J=1,JJ
08762       IF(A(I,J))26,27,26
08878    26 A(III,J)=A(III,J)-A(I,J)
09094    27 CONTINUE
09130       GO TO 23
09138    40 K=III
09162    44 J=0
09186       W(K)=0.0
09234       L(K)=0
09282    42 J=J+1
09318       IF(J-JJ)41,45,45
09386    41 IF(A(K,J))43,42,42
09502    43 IF(W(K)-A(K,J))42,42,47
09654    47 W(K)=A(K,J)
09762       L(K)=J
09810       GO TO 42
09818    45 IF(L(K))46,62,46
09898    46 KJ=L(K)
09946       DO 120 I=2,II
09958       IF(A(I,KJ))120,120,121
10074   120 CONTINUE
10110       PRINT 130
10134       PAUSE
10146   130 FORMAT(9HUNBOUNDED)
10188   121 I=1
10212       JK=0
10236    50 I=I+1
10272       IF(I-II)52,52,56
10340    52 IF(A(I,KJ))50,50,51
10456    51 X=A(I,JJ)/A(I,KJ)
10612       IF(JK)55,53,55
10668    55 IF(X-XMIN)53,50,50
10736    53 XMIN=X
10760       JK=I
10784       GO TO 50
10792    56 X=A(JK,KJ)
10876       L(JK)=KJ
10924       DO 57 I=1,III
```

SOLUTION PHASE (CONTINUED)

```
10936     57 W(I)=A(I,KJ)
11080        IJ=JK-1
11116        DO 59 I=1,IJ
11128        DO 59 J=1,JJ
11140        IF(A(JK,J))58,59,58
11256     58 IF(W(I))580,59,580
11336    580 A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
11612     59 CONTINUE
11684        IJ=JK+1
11720        DO 61 I=IJ,III
11732        DO 61 J=1,JJ
11744        IF(A(JK,J))60,61,60
11860     60 IF(W(I))600,61,600
11940    600 A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
12216     61 CONTINUE
12288        DO 205 J=1,JJ
12300    205 A(JK,J)=A(JK,J)/X
12492        KKK=KKK+1
12528        PRINT 105,KKK,A(K,JJ),L(JK)
12660        IF(SENSE SWITCH 4)70,44
12680     62 IF(K-1)70,70,63
12748     63 IJ=JJ-1
12784        DO 65 J=1,IJ
12796        IF(A(K,J)-.0001)65,65,66
12924     65 CONTINUE
12960        PRINT 103
12984        DO 130 J=1,JJ
12996    130 A(III,J)=0.0
13080    103 FORMAT(8HFEASABLE)
13120        K=1
13144        KKK=0
13168        GO TO 44
13176     66 PRINT 6
13200        PAUSE
13212     70 PRINT 8,A(1,JJ)
13284        PRINT 7
13308        DO 71 I=2,II
13320     71 PRINT 5,L(I),A(I,JJ)
13476        IF(SENSE SWITCH 3)72,75
13496     72 PRINT 9
13520        IJ=JJ-1
13556        DO 74 J=1,IJ
13568        IF(A(1,J))73,74,73
13672     73 PRINT 5,J,A(1,J)
13756     74 CONTINUE
13792     75 IF(SENSE SWITCH 2)76,80
13812     76 IJ=JJ-1
13848        DO 77 J=1,IJ
13860        DO 77 I=1,III
13872        IF(A(I,J))78,77,78
13988     78 PUNCH 2,I,J,A(I,J)
14096     77 CONTINUE
14168        PUNCH 100
14192        DO 90 I=1,III
14204     90 PUNCH 2,I,L(I),A(I,JJ)
14372     80 PAUSE
14384        END
```


RIGHT HAND SIDE CHANGER PHASE

```
08000        DIMENSION A(44,96),W(44),L(44),II(1),JJ(1),III(1),X(1)
08000        DIMENSION JK(1),KJ(1),KKK(1),IJ(1),K(1),I(1),J(1),XMIN(1)
08000      1 FORMAT(I4,I4,F10.5,F10.5)
08038      2 FORMAT(10HINFEASABLE)
08082      3 FORMAT(33HITER       FUNCTIONAL       VAR. IN)
08172    100 FORMAT(F20.5,F20.5)
08200    101 FORMAT(32H       OLD RHS              NEW RHS)
08288    104 FORMAT(I4,F15.2,10X,I4)
08348    105 FORMAT(19HVARIABLE       VALUE)
08410    106 FORMAT(I4,F20.8)
08438    107 FORMAT(10HFUNCTIONAL,F20.8)
08488    109 FORMAT(23HVARIABLE       SHAD. COST)
08558    110 FORMAT(I4,I4,F20.8)
08590    111 FORMAT(1H )
08616      8 PRINT 101
08640        K=1
08664        KKK=0
08688      4 READ 1,KJ,JK,W(1),W(III)
08772        IF(KJ)81,83,81
08828     83 IF(JK)84,11,84
08884     81 IJ=1
08908        GO TO 5
08916     84 IJ=2
08940      5 PRINT 100,W(1),W(III)
09000        W(1)=W(III)-W(1)
09060        DO 10 I=IJ,II
09072     10 A(I,JJ)=A(I,JJ)+W(1)*A(I,JK)
09336        GO TO 4
09344     11 PRINT 3
09368     15 XMIN=0.0
09392        L(K)=0
09440        I=1
09464     18 I=I+1
09500        IF(I-II)16,16,20
09568     16 IF(A(I,JJ))17,18,18
09684     17 IF(A(I,JJ)-XMIN)19,18,18
09812     19 XMIN=A(I,JJ)
09896        L(K)=I
09944        GO TO 18
09952     20 IF(L(K))21,70,21
10032     21 JK=L(K)
10080        J=0
10104        XMIN=0.0
10128        L(K)=0
10176     26 J=J+1
10212        IF(J-JJ)23,24,24
10280     23 IF(A(JK,J))25,26,26
10396     25 IF(A(1,J))26,26,7
10500      7 X=A(1,J)/A(JK,J)
10644        IF(L(K))6,27,6
10724      6 IF(X-XMIN)26,26,27
10792     27 XMIN=X
10816        L(K)=J
10864        GO TO 26
10872     24 IF(L(K))31,30,31
```

RIGHT HAND SIDE CHANGER PHASE (CONTINUED)

```
10952    30 PRINT 2
10976       PAUSE
10988    31 KJ=L(K)
11036       L(JK)=KJ
11084       X=A(JK,KJ)
11168       DO 33 I=1,III
11180    33 W(I)=A(I,KJ)
11324       IJ=JK-1
11360       DO 35 I=1,IJ
11372       DO 35 J=1,JJ
11384       IF(A(JK,J))34,35,34
11500    34 IF(W(I))37,35,37
11580    37 A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
11856    35 CONTINUE
11928       IJ=JK+1
11964       DO 40 I=IJ,III
11976       DO 40 J=1,JJ
11988       IF(A(JK,J))38,40,38
12104    38 IF(W(I))39,40,39
12184    39 A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
12460    40 CONTINUE
12532       DO 50 J=1,JJ
12544    50 A(JK,J)=A(JK,J)/X
12736       KKK=KKK+1
12772       PRINT 104,KKK,A(1,JJ),L(JK)
12892       GO TO 15
12900    70 PRINT 107,A(1,JJ)
12972       PRINT 105
12996       DO 71 I=2,II
13008    71 PRINT 106,L(I),A(I,JJ)
13164       IF(SENSE SWITCH 3)72,75
13184    72 PRINT 109
13208       IJ=JJ-1
13244       DO 74 J=1,IJ
13256       IF(A(1,J))73,74,73
13360    73 PRINT 106,J,A(1,J)
13444    74 CONTINUE
13480    75 IF(SENSE SWITCH 2)76,80
13500    76 IJ=JJ-1
13536       DO 77 J=1,IJ
13548       DO 77 I=1,III
13560       IF(A(I,J))78,77,78
13676    78 PUNCH 110,I,J,A(I,J)
13784    77 CONTINUE
13856       PUNCH 111
13880       DO 90 I=1,II
13892    90 PUNCH 110,I,L(I),A(I,JJ)
14060    80 PAUSE
14072       GO TO 8
14080       END
```

COMPUTER
TECHNOLOGY