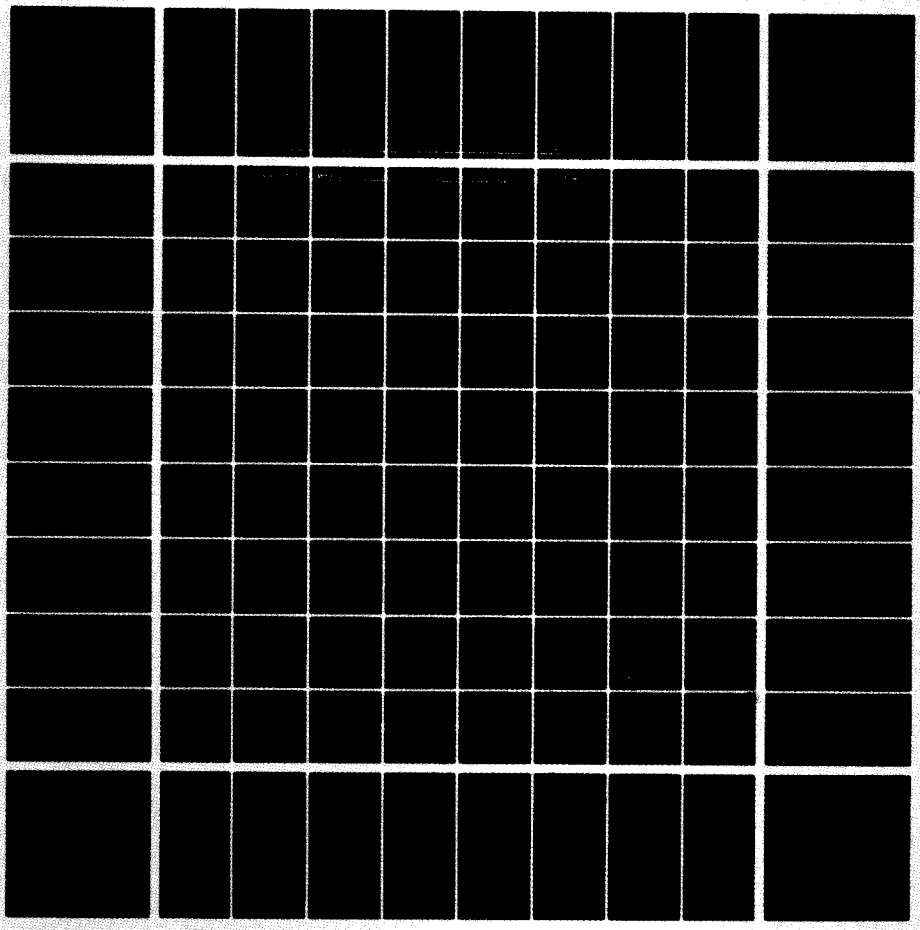
HEWLETT-PACKARD

HP-41C

STANDARD APPLICATIONS



NOTICE

The program material contained herein is supplied without representation or warranty of any kind. Hewlett-Packard Company therefore assumes no responsibility and shall have no liability, consequential or otherwise, of any kind arising from the use of this program material or any part thereof.

HP-41C

Standard Applications Handbook

May 1981

00041-90366

Printed in U.S.A.

INTRODUCTION

This applications handbook contains a collection of programs that demonstrate the power and versatility of your HP-41C in programmed problem-solving. You will find the programs useful, entertaining, and fascinating. By entering and executing them, you'll get an immediate "hands-on" glimpse of the advanced capabilities of your HP-41C, and—thanks to its Continuous Memory —you'll have them available in the future ready to use.

Studying all of these professionally designed programs will help you develop your own programming expertise. The benefits of owning an HP-41C can be realized through the imaginative exploitation of its programming power and versatility, which enable you to customize your HP-41C to suit your particular needs.

For each of the 10 programs in this handbook we've included a description, instructions, one or more example problems, program highlights, and a program listing. Before entering any of the programs, take a few minutes to study the sections Keying a Program Into the HP-41C and Format of User Instructions at the front of this handbook. You might understand them better and learn a lot more from them if you've first read through the *HP-41C Owner's Handbook and Programming Guide.*

When you've selected a program you'd like to execute, key it in by following the program listing, then refer to the table of instructions for detailed information on how to use the program. You'll probably need to refer to these instructions only the first few times you run the program. Afterwards, the program's prompting should provide the necessary instructions, including which data should be input, the keys to press, and the kind of output.

The Program Highlights present programming techniques of particular interest. Studying them will help you understand the operation of parts of the program, and you may find uses for them as part of programs you write yourself. For an in-depth understanding of the program's operation, and to learn more about efficient and versatile programming techniques, also study the comments included in the program listings.

Except for the blackjack game, all programs in this handbook can be keyed into the basic HP-41C. The blackjack game requires one additional memory module. As you expand your HP-41C system, you will find that some of these programs work well as a basis for larger programs of your own. You might want to modify some programs slightly to suit your individual needs—that's the beauty of programmability.

CONTENTS

FORMAT OF USER INSTRUCTIONS

The User Instructions which accompany each program are your guide to operating the programs in this handbook.

The form is composed of five labeled columns. Reading from left to right, the first column, labeled STEP, gives the instruction step number.

The INSTRUCTIONS column gives instructions and comments concerning the operations to be performed.

The INPUT column specifies the input data, the units of data if applicable, or the appropriate alpha response to a prompted question. Data Input keys consist of 0 to 9 and the decimal point (the numeric keys), [EEX] (enter exponent), and [CHS] (change sign).

The FUNCTION column specifies the keys to be pressed after keying in the corresponding input data.

Whenever a statement in the INPUT or FUNCTION column is printed in gold, the ALPHA mode must be on before the statement can be keyed in. For example, [XEQ] A4C means press the following keys: [XEQ] [ALPHA] A■4C [ALPHA]. Of course, you could assign the function A4C to any key you chose by pressing ■ [ASN] [ALPHA] A■4C [ALPHA] [KEY]. Then you could simply press [KEY] in USER mode to execute the function.

The DISPLAY column specifies prompts as well as intermediate and final answers and (where applicable) their units.

Above the DISPLAY column is a box which specifies the SIZE or minimum number of data registers used by the program. Program memory should be SIZEd before keying in the program or it might not fit. Refer to pages 73 and 117 in the Owner's Handbook for a complete description of how to size calculator memory.

KEYING A PROGRAM INTO THE HP-41C

There are several things that you should keep in mind while you are keying in programs from the program listings provided in this book. The output from the HP 82143A printer provides a convenient way of listing and an easily understood method of keying in programs without showing every keystroke. This type of output is what appears in this handbook. Once you understand the procedure for keying programs in from the printed listings, you will find this method simple and fast. Here is the procedure:

1. At the end of each program listing is a listing of status information required to properly execute that program. Included is the SIZE allocation required. Before you begin keying in the program, press [XEQ] [ALPHA] SIZE [ALPHA] and specify the allocation (three digits; e.g., 10 should be specified as 010).

   Also included in the status information is the display format and status of flags important to the program. To ensure proper execution, check to see that the display status of the HP-41C is set as specified and check to see that all applicable flags are set or clear as specified.

2. Set the HP-41C to PRGM mode (press the [PRGM] key) and press [■] [GTO] [.] [.] to prepare the calculator for the new program.

3. Begin keying in the program. Following is a list of hints that will help you when you key in your programs from the program listings in this handbook.

   a. When you see " (quote marks) around a character or group of characters in the program listing, those characters are ALPHA. To key them in, simply press [ALPHA], key in the characters, then press [ALPHA] again. So 06 "SAMPLE" would be keyed in as [ALPHA] SAMPLE [ALPHA].

   b. The diamond in front of each LBL instruction is only a visual aid to help you locate labels in the program listings. When you key in a program, ignore the diamond.

   c. The printer indication of the divide sign is /. When you see / in the program listing, press [÷].

   d. The printer indication of the multiply sign is *. When you see * in the program listing, press [×].

   e. The ⊢ character in the program listing is an indication of the [APPEND] function. When you see ⊢, press [■] [APPEND] in ALPHA mode (press [■] and the K key).

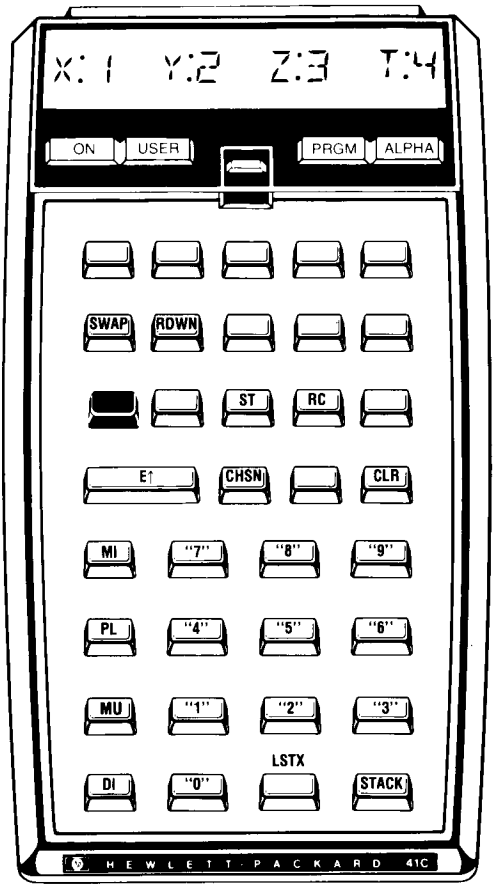
f. All operations requiring register addresses accept those addresses in these forms:
nn (a two-digit number)
IND nn (INDIRECT: [■], followed by a two-digit number)
X, Y, Z, T, or L (a STACK address: [▪] followed by X, Y, Z, T, or L)
IND X, Y, Z, T, or L (INDIRECT stack: [■][▪] followed by X, Y, Z, T, or L)

Indirect addresses are specified by pressing [■] and then the indirect address. Stack addresses are specified by pressing [▪] followed by X, Y, Z, T, or L. Indirect stack addresses are specified by pressing [■][▪] and X, Y, Z, T, or L.

Printer Listing	Keystrokes	Display
01◆LBL "SAMPLE"	[■] [LBL] [ALPHA] SAMPLE [ALPHA]	*01 LBL ⊤ SAMPLE*
02 "THIS IS A "	[ALPHA] THIS IS A [ALPHA]	*02 ⊤THIS IS A*
03 "⊢SAMPLE"	[ALPHA] [■] [APPEND] SAMPLE	*03 ⊤⊢ SAMPLE*
04 AVIEW	[■] AVIEW [ALPHA]	*04 AVIEW*
05 6	6	*05 6*
06 ENTER↑	[ENTER↑]	*06 ENTER ↗*
07 -2	2 [CHS]	*07 -2*
08 /	[÷]	*08 /*
09 ABS	[XEQ] [ALPHA] ABS [ALPHA]	*09 ABS*
10 STO IND L	[STO] [■] [▪] L	*10 STO IND L*
11 "R3="	[ALPHA] R3= [■] [ARCL] 03	*11 ⊤ R3=*
12 ARCL 03	[■] [AVIEW]	*12 ARCL 03*
13 AVIEW	[ALPHA]	*13 AVIEW*
14 RTN	[■] [RTN]	*14 RTN*

RPN PRIMER

This program is an aid to understanding and using RPN, the logic system used in the HP-41C. All four registers of the operational stack are visible simultaneously so that the effect of a given keystroke sequence can be seen rather than inferred. The functions provided, assigned as shown in the instructions, appear on the keyboard below. These functions all exit to a routine which displays the operational stack. It is possible to observe the effect on the stack of functions which are not included within this program. Simply execute the desired function, then press the [R/S] key, to which STACK is assigned. The only operational differences between this redefined calculator and the actual one are that only single-digit numbers can be keyed in and that STO/RCL address only a single register (thus requiring no address).



SIZE: 001

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Set status and key in the the program			
2	Assign * its routines as shown and select USER mode. These suggested assignments result in the keyboard shown on the previous page.			
3	Press desired keystroke sequence and watch stack contents change			
4	The functions RUP and CLSTK are obtained by and (or you could assign these functions as well)		XEQ RUP XEQ CLSTK	
	* To assign a function, say FCN, to a key, say the √x key,		ASN ALPHA FCN ALPHA √x	

SWAP x⇄y
ST STO RDWN R↓
E↑ ENTER↑ RC RCL
CLR ← CHSN CHS
PL + MI −
MU × DI ÷
9 9 8 8
7 7 6 6
5 5 4 4
3 3 2 2
1 1 0 0
LSTX LASTx STACK R/S

Example 1:

Evaluate the expression

$$\frac{(2 + b)\, b}{8 - b}$$

for $b = 3$

Keystrokes:

Function	Display
XEQ ALPHA CLSTK ALPHA	*X:0 Y:0 Z:0 T:0*
2	*X:2 Y:0 Z:0 T:0*
ENTER↑	*X:2 Y:2 Z:0 T:0*

3	*X:3 Y:2 Z:0 T:0*
[+]	*X:5 Y:0 Z:0 T:0*
[■] [LASTX]	*X:3 Y:5 Z:0 T:0*
[×]	*X:15 Y:0 Z:0 T:0*
8	*X:8 Y:15 Z:0 T:0*
[■] [LASTX]	*X:3 Y:8 Z:15 T:0*
[−]	*X:5 Y:15 Z:0 T:0*
[+]	*X:3 Y:0 Z:0 T:0*

Example 2:

Without disturbing the above results, compute

$$\frac{2 + 4\,(9 - 7)}{6 - 4}$$

Function	**Display**	
9	*X:9 Y:3 Z:0 T:0*	
[ENTER↑]	*X:9 Y:9 Z:3 T:0*	After an [ENTER↑], the stack does not lift when new data is keyed in
7	*X:7 Y:9 Z:3 T:0*	
[−]	*X:2 Y:3 Z:0 T:0*	
4	*X:4 Y:2 Z:3 T:0*	
[×]	*X:8 Y:3 Z:0 T:0*	
2	*X:2 Y:8 Z:3 T:0*	
[+]	*X:10 Y:3 Z:0 T:0*	
6	*X:6 Y:10 Z:3 T:0*	
[ENTER↑]	*X:6 Y:6 Z:10 T:3*	
4	*X:4 Y:6 Z:10 T:3*	
[−]	*X:2 Y:10 Z:3 T:3*	
[+]	*X:5 Y:3 Z:3 T:3*	Notice that the answer remaining from Example 1 did not cause a difficulty in Example 2

Example 3:

Convert the complex number 3 + 4i to polar form.

4	***X:4 Y:5 Z:3 T:3***	
[ENTER↑]	***X:4 Y:4 Z:5 T:3***	
3	***X:3 Y:4 Z:5 T:3***	
[■] [R-P]	***5***	
STACK	***X:5 Y:53 Z:5 T:3***	Remember that STACK is assigned to [R/S]

Programming Highlight

What is especially useful in this program is the display routine STACK. You might like to keep it handy to view the entire stack from time to time as you solve your own problems.

```
 01*LBL "CLS
TK"
 02 CLST
 03 GTO 14
 04*LBL "1"
 05 FS?C 05
 06 CLX
 07 1
 08 GTO 14
 09*LBL "2"
 10 FS?C 05
 11 CLX
 12 2
 13 GTO 14
 14*LBL "3"
 15 FS?C 05
 16 CLX
 17 3
 18 GTO 14
 19*LBL "4"
 20 FS?C 05
 21 CLX
 22 4
 23 GTO 14
 24*LBL "5"
 25 FS?C 05
 26 CLX
 27 5
 28 GTO 14
 29*LBL "6"
 30 FS?C 05
 31 CLX
 32 6
 33 GTO 14
 34*LBL "7"
 35 FS?C 05
 36 CLX
 37 7
 38 GTO 14
 39*LBL "8"
 40 FS?C 05
 41 CLX
 42 8
 43 GTO 14
 44*LBL "9"
 45 FS?C 05
 46 CLX
 47 9
 48 GTO 14
 49*LBL "0"
 50 FS?C 05
 51 CLX
 52 0
 53 GTO 14
 54*LBL 13
 55 CF 05
 56*LBL 14
 57*LBL "STA
CK"
 58 "X:"
 59 ARCL X
 60 "⊢ Y:"
 61 ARCL Y
 62 "⊢ Z:"
 63 ARCL Z
 64 "⊢ T:"
 65 ARCL T
 66 AVIEW
 67 RTN
 68*LBL "E↑"
 69 SF 05
 70 ENTER↑
 71 GTO 14
 72*LBL "RDW
N"
 73 RDN
 74 GTO 13
 75*LBL "SWA
P"
 76 X<>Y
 77 GTO 14
 78*LBL "RUP
"
 79 R↑
 80 GTO 13
 81*LBL "PL"
 82 +
 83 GTO 13
 84*LBL "MI"
 85 -
 86 GTO 13
 87*LBL "MU"
 88 *
 89 GTO 13
 90*LBL "DI"
 91 /
 92 GTO 13
 93*LBL "CLR
"
 94 SF 05
```

Clear stack.

If lift disabled clear x first
Input a 1.

See note

Input a 2.

Input a 3.

Input a 4.

Input a 5.

Input a 6.

Input a 7.

Input an 8.

Input a 9.

Input a 0.

Enable stack lift.

Display stack.

Disable stack lift.

Roll down.

Swap x and y.

Roll up.

Plus.

Minus.

Multiply.

Divide.

R00 Storage

95 CLX 96 GTO 14 97◆LBL "CHS N" 98 CHS 99 GTO 14 100◆LBL "ST" 101 STO 00 102 GTO 14 103◆LBL "RC" 104 FS?C 05 105 CLX 106 RCL 00 107 GTO 14 108◆LBL "LST X" 109 FS?C 05 110 CLX 111 LASTX 112 GTO 14	Disable stack lift and clear x. Change sign. Store. If lift disabled clear x first. Recall. This step need not be keyed in.		
Important Status Size = 001 Fix 0 Flags used F05 Set = Stack lift disable F29 Clear for no radix point			

Note: You will find it convenient to assign FS?C to some key, for example ■ [ASN] [ALPHA] FS?C [ALPHA] [LN] assigns FS?C to the [LN] key. You can then press [LN] once to get FS?C__ __ in the display and a second time to create FS?C 05. Remember that you must be in USER mode or you will get two LN's instead.

CALENDAR FUNCTIONS

This program provides an interchangeable solution of dates and days between dates. Given two dates, the program can determine the number of days between them, or it can compute a second date from a first one and a number of days. Dates are input in the form mm.ddyyyy. They are output as MONTH dd,yyyy.

Another feature of this program is that it can convert a date to its day of the week, displaying the result with the correct day name.

This program is valid from March 1, 1900 to February 28, 2100. The program does not check input data. Thus, if an improper format or an invalid date (i.e., February 30) is keyed in, erroneous answers will result.

				SIZE: 010
STEP	**INSTRUCTIONS**	**INPUT**	**FUNCTION**	**DISPLAY**
1	Set status, key in the program and select USER mode			
	DAY OF THE WEEK			
2a	Input date and calculate day	DATE*	[E]	Day of Week
3a	Repeat step 2a for a new date			
	DAYS BETWEEN DATES			
2b	Input two of the following:			
	First date	D 1*	[A]	Date 1*
	Second date	D 2*	[B]	Date 2*
	Days between dates	D	[C]	D
3b	Calculate one of the following:			
	First date		[A]	Date 1*
	Second date		[B]	Date 2*
	Days between dates		[C]	D
4	Repeat step 2b for new data (values which do not change need not be re-entered)			
	* Dates are input in the form mm.ddyyyy; they are output in the form MONTH dd,yyyy.			

Example 1:

On what day of the week was February 19, 1946?

Keystrokes:	**Display:**
2.191946[E]	***TUESDAY***

Example 2:

What date is 10,000 days after August 4, 1978?

Keystrokes:	**Display:**
8.041978 A 10000 C B	***DEC 20,2005***

Example 3:

A man born on December 18, 1913, is the father of a boy born on February 19, 1946. On what date will the father be twice as many days old as his son?

Keystrokes:	**Display:**	
12.181913 A	***DEC 18,1913***	
2.191946 B	***FEB 19,1946***	
C	***11751***	Number of days.
2 × C B	***APR 23,1978***	Twice as many days after Date 1.

Programming Highlight

This program utilizes the "selectable radix point" feature of the HP-41C to format its date display. With a date of the form mm.ddyyyy in the x-register, XEQ IND X executes a subroutine which places the three-letter month designation in the alpha-register. The program then multiplies the fractional part of X by 100, clears the decimal point flag, and appends the day and year to the alpha display. Thus an original x-value of 12.251978 yields a display of DEC 25,1978.

Note: Because of its length, this program was written using only local labels. If the program pointer should ever point to somewhere else in memory, you can move it back using CAT 1 as described on page 140 of your Owner's Handbook.

```
01◆LBL A
02 RCL 04
03 RCL 01
04 -
05 3
06 GTO 20
07◆LBL B
08 RCL 03
09 RCL 01
10 +
11 4
12◆LBL 20
13 STO 02
14 RDN
15 365.25
16 STO 05
17 30.6001
18 STO 06
19 RDN
20 RDN
21 FS?C 22
22 GTO 21
23 STO IND
02
24 122.1
25 -
26 RCL 05
27 /
28 INT
29 STO 09
30 RCL 05
31 *
32 INT
33 RCL IND
02
34 -
35 CHS
36 STO 00
37 RCL 06
38 /
39 INT
40 STO 07
41 RCL 00
42 X<>Y
43 RCL 06
44 *
45 INT
46 -
47 STO 08
48 RCL 07
49 1
50 RCL 08
51 %
```

Calculate Δ days and put control 3 in display.

Calculate Δ days and put control 4 in display.

Store control code.

Store constants.

Return Δ days to display.

Store Δ days according to control code.

Calculate day of month.

```
52 -
53 -
54 RCL 07
55 14
56 /
57 XEQ 22
58 RCL 09
59 1 E6
60 /
61 +
62 GTO 25
63◆LBL 21
64 RDN
65 FC? 06
66 STO IND
02
67 ENTER↑
68 INT
69 STO 07
70 -
71 1 E2
72 *
73 ENTER↑
74 INT
75 STO 08
76 -
77 1 E4
78 *
79 STO 09
80 RCL 07
81 1
82 +
83 ENTER↑
84 1/X
85 .7
86 +
87 CHS
88 XEQ 22
89 RCL 06
90 *
91 INT
92 RCL 09
93 RCL 05
94 *
95 INT
96 +
97 RCL 08
98 +
99 X<> IND
02
100 FS?C 06
101 RTN
102◆LBL 25
```

Break date input into the individual components of mm,dd,yyyy.

Compute day number.

R00 = Scratch
R01 = Δdays
R02 = Pointer
R03 = Day #1
R04 = Day #2

R05 = 365.25
R06 = 30.600
R07 = m
R08 = d
R09 = y

```
103 ENTER↑
104 XEQ IND
X
105 FRC
106 1 E2
107 *
108 CF 28
109 FIX 4
110 ARCL X
111 RDN
112 AVIEW
113 SF 28
114 RTN
115*LBL 22
116 INT
117 ST+ 09
118 12
119 *
120 -
121 RTN
122*LBL C
123 CF 29
124 FIX 0
125 STO 01
126 FS?C 22
127 RTN
128 RCL 04
129 RCL 03
130 -
131 STO 01
132 RTN
133*LBL E
134 SF 06
135 SF 22
136 RCL 05
137 5
138 XEQ 20
139 RCL IND
02
140 7
141 MOD
142 13
143 +
144 XEQ IND
X
145 AVIEW
146 RTN
147*LBL 13
148 "FRIDAY"
149 RTN
150*LBL 14
151 "SATURDA
Y"
152 RTN
```

Compute day of week.

```
153*LBL 15
154 "SUNDAY"
155 RTN
156*LBL 16
157 "MONDAY"
158 RTN
159*LBL 17
160 "TUESDAY
"
161 RTN
162*LBL 18
163 "WEDNESD
AY"
164 RTN
165*LBL 19
166 "THURSDA
Y"
167 RTN
168*LBL 01
169 "JAN "
170 RTN
171*LBL 02
172 "FEB "
173 RTN
174*LBL 03
175 "MAR "
176 RTN
177*LBL 04
178 "APR "
179 RTN
180*LBL 05
181 "MAY "
182 RTN
183*LBL 06
184 "JUN "
185 RTN
186*LBL 07
187 "JUL "
188 RTN
189*LBL 08
190 "AUG "
191 RTN
192*LBL 09
193 "SEP "
194 RTN
195*LBL 10
196 "OCT "
197 RTN
198*LBL 11
199 "NOV "
200 RTN
201*LBL 12
202 "DEC "
```

Important Status
Size = 010
Fix 4

Flags used
F06
F22
F28
F29

WORD GUESSING GAME

This program is a version of the word game "hangman." The first player makes up a six-character word and gives it to the calculator. The second player guesses various letters until he has completed the word. After each guess, the calculator displays all correctly guessed characters in their appropriate places. When the entire word has been guessed, the number of guesses is displayed.

				SIZE: 019
STEP	**INSTRUCTIONS**	**INPUT**	**FUNCTION**	**DISPLAY**
1	Set status and key in the program.			
2	Begin running the program		XEQ WORDS	KEY IN WORD
3	First player: Key in your word	any of six characters	R/S	LETTER?
4	Second player: Guess a character	any character	R/S	word so far LETTER?
5	Repeat step 4 to guess more characters. When word is complete, you will see DONE, WORD IS <word>, and YOU TOOK nn GUESSES.			

Example:

Hide "HP-41C" and then guess it.

Keystrokes:	**Display:**	
XEQ ALPHA WORDS ALPHA	***KEY IN WORD***	
HP-41C R/S	***LETTER?***	(Notice that the program stops in ALPHA mode.)
A R/S	***LETTER?***	
P R/S	***P*** ***LETTER?***	
C R/S	***P C*** ***LETTER?***	
H R/S	***HP C*** ***LETTER?***	
■ 4 R/S	***HP 4 C*** ***LETTER?***	

■ 1 R/S

■ – R/S

```
HP 41C
LETTER?
DONE
WORD IS <HP-41C>
YOU TOOK 7 GUESSES
```

Programming Highlight

Two special routines were used while developing this program: SPEL and DESPEL. Their function was to build up a word from a collection of letters and to take apart a word into its component letters. Only DESPEL remains in the final program because the job performed by SPEL was already done by the letter-comparison portion of the program.

A code must be passed through the x-register to SPEL and DESPEL. This code tells SPEL where to find its letters, DESPEL, where to put its letters. The code is of the form

fl.0ll for SPEL or *ll.0ff* for DESPEL

where

fl = register for first letter
ll = register for last letter

ff = *fl* − 1

SPEL and DESPEL or other similar routines may be used to encode and decode many types of strings. A similar routine was used in the hexadecimal conversion program (page 28).

```
01♦LBL "SPE
L"
02 STO 07
03♦LBL 08
04 ARCL IND
07
05 ISG 07
06 GTO 08
07 RTN
```

Assumes a cleared ALPHA register.
Store the counter *fl.0ll*.
Build the word.
If not last letter, then repeat loop.

```
01♦LBL "DES
PEL"
02 STO 07
03 ASTO 00
04♦LBL 07
05 " "
06 ARCL 00
07 ASTO 00
08 ASHF
09 ASTO IND
07
10 DSE 07
11 GTO 07
12 RTN
```

Store the counter *ll.0ff*.
Save the word.
Save all but the last letter.
Save the last letter.
If not all letters, then repeat loop.

```
 01♦LBL "WOR
DS"
 02 "KEY IN
WORD"
 03 AON
 04 PROMPT
 05 ASTO 08
 06 6
 07 XEQ "DES
PEL"
 08 .9
 09 STO 17
 10 "      "
 11 ASTO 09
 12 16.01
 13 XEQ "DES
PEL"
 14♦LBL "LTT
R"
 15 CLA
 16 ASTO 09
 17 "LETTER?
"
 18 AON
 19 PROMPT
 20 ASTO 10
 21 ISG 17
 22 1.006
 23 STO 18
 24♦LBL 06
 25 " "
 26 ASTO Y
 27 RCL 18
 28 10
 29 +
 30 CLA
 31 ARCL IND
 X
 32 RDN
 33 ASTO X
 34 X≠Y?
 35 GTO 00
 36 CLA
 37 ARCL 10
 38 ASTO Y
 39 CLA
 40 ARCL IND
 18
 41 ASTO X
 42 X=Y?
```

Store secret word. Place letters in R01 to to R06

6 spaces.

Place blanks in R11 to R16.

Ask player for letter.

Save letter.
Count # letters.
Initialize counter.
Begin loop 6.

If position already has letter, then display it.

If guess is correct

```
 43 GTO 00
 44 " "
 45 ASTO X
 46♦LBL 00
 47 CLA
 48 ARCL 09
 49 ARCL X
 50 ASTO 09
 51 AVIEW
 52 10
 53 RCL 18
 54 +
 55 CLA
 56 ARCL Y
 57 ASTO IND
 X
 58 ISG 18
 59 GTO 06
 60 CLA
 61 ARCL 08
 62 ASTO Y
 63 CLA
 64 ARCL 09
 65 ASTO X
 66 X=Y?
 67 GTO 00
 68 PSE
 69 PSE
 70 GTO "LTT
R"
 71♦LBL 00
 72 "DONE"
 73 AVIEW
 74 "WORD IS
 <"
 75 ARCL 09
 76 "├>"
 77 AVIEW
 78 PSE
 79 PSE
 80 RCL 17
 81 INT
 82 "YOU TOO
K "
 83 ARCL X
 84 "├ GUESS
ES"
 85 AVIEW
 86 RTN
 87♦LBL "DES
PEL"
```

Then display i. Else display blank.

Add a letter to the display.

Repeat loop six times.

If words are same, then done.
Else
ask for another guess.

Display word.

Display #guesses.

R00 = Temporary
R01 = 1st letter, SW
R02 = 2nd letter, SW
R03 = 3rd letter, SW
R04 = 4th letter, SW
R05 = 5th letter, SW
R06 = 6th letter, SW
R07 = Counter
R08 = Secret word, (SW)
R09 = Player's word, (PW)
R10 = Current letter
R11 = 1st letter, PW
R12 = 2nd letter, PW
R13 = 3rd letter, PW

```
88 STO 07
89 ASTO 00
90◆LBL 07
91 " "
92 ARCL 00
93 ASTO 00
94 ASHF
95 ASTO IND
07
96 DSE 07
97 GTO 07
98 RTN
```

Subroutine to separate a word into its letters.

Important Status
Size = 019
Fix 0
CF 29

Flags used
F29 Clear to suppress decimal point

R14 = 4th letter, PW
R15 = 5th letter, PW
R16 = 6th letter, PW
R17 = Counter
R18 = Counter

ARITHMETIC TEACHER

This program generates arithmetic practice problems. You may choose the maximum values of the numbers used and whether the problems are addition, subtraction, multiplication or division. After 10 problems have been worked, a percentage score is displayed.

The program can be started by [XEQ] [ALPHA] TEACH [ALPHA] . The calculator prompts for the largest number to use in the problems. After keying in the maximum number and pressing [R/S] , you will see a display of "+ , − ,×, /?" with the ALPHA annunciator turned on. Simply press the gold shift key, one of the arithmetic functions, and [R/S] to begin the exercise. ALPHA mode will be turned off automatically.

After each problem is presented, key in your answer and press [R/S] . A correct answer is rewarded with ***YES*** and a new problem is presented. An incorrect answer elicits an unpleasant sound and the message ***NO***, and you are given a second chance. The machine tells you the answer if you make two mistakes on the same problem, then it continues with a new one. If all 10 were worked correctly the first time, a fanfare is played. The program then begins again with the "+ , − , ×, /?" question.

The series of problems is determined by a seed (number) between 0 and 1 that is in the X-register when you begin the program. If you want to repeat a particular series of problems, key in the same seed each time. If no seed is keyed in, the program simply uses the number already in the X-register.

Reference: Knuth, *The Art of Computer Programming,* Addison Wesley, Reading, Mass., 1978.

				SIZE: 010
STEP	**INSTRUCTIONS**	**INPUT**	**FUNCTION**	**DISPLAY**
1	Set status and key in the program			
2	Input a seed ($0 \leq$ seed < 1) and begin program.	seed	[XEQ] TEACH	MAX NUMBER?
3	Input the largest number to use	N	[R/S]	+, −, ×, /?
4	Select			equation callouts
	addition	+	[R/S]	$(n_1) + (n_2) = ?$
	subtraction	−	[R/S]	$(n_1) - (n_2) = ?$
	multiplication	×	[R/S]	$(n_1) \times (n_2) = ?$
	division	/	[R/S]	$(n_1) / (n_2) = ?$
5	Key in your answer.	answer	[R/S]	YES or NO
6	After 10 problems have been worked, your score is displayed and you may continue at step 4.			(SCORE)% RIGHT

Example:

Using a seed of .021946, do some subtraction problems with arguments up to 14.

Keystrokes:	**Display:**
.021946	
XEQ ALPHA TEACH ALPHA	***MAX NUMBER?***
14 R/S	*+, −, ×, /?*
■ − R/S	***12−5=?***
7 R/S	***YES***
	14−13=?
1 R/S	***YES***
	13−6=?
8 R/S	***NO 13−6=?***
7 R/S	***YES***
	14−11=?
3 R/S	***YES***
	14−7=?
6 R/S	***NO 14−7=?***
8 R/S	***NO 14−7=7***
	13−2=?
11 R/S	***YES***
	14−13=?
1 R/S	***YES***
	14−10=?
4 R/S	***YES***
	12−9=?
3 R/S	***YES***
	14−10=?
4 R/S	***YES***
	90% RIGHT
	+, −, ×, /?

Programming Highlight

This program uses a combination of the HP-41C's alpha capabilities: indirect subroutine calls together with output labels consisting of user-supplied alpha characters.

At one point in the program, you are asked to key in a +, −, ×, or / symbol depending on which type of problem you wish to work. The program stores this symbol in register 06, generates two numbers, and then executes the subroutine whose name was stored in R_{06}. That same symbol is then recalled to help create the display showing the problem you must work.

Another interesting portion of this program is the random number generator:

$$r_{n+1} = \text{FRC}\ (9821 \times r_n + .211327)$$

This generator was developed by Don Malm as part of an HP-65 Users' Library program. It passes the spectral test (Knuth, V.2, § 3.4) and, because its parameters satisfy Theorem A (op. cit., p. 15), it generates one million distinct random numbers between 0 and 1 regardless of the value selected for r_0.

Because the basic random number generator delivers numbers between 0 and 1, it is necessary to do further manipulation of the random numbers to get the integers required for the arithmetic problems. By multiplying the random numbers by an integer N, then taking the integer part, numbers from 0 to N-1 may be generated. This program uses your maximum desired number plus 1 to generate numbers from 0 to your desired maximum.

```
 01♦LBL "TEA
CH"
 02 CF 29
 03 FIX 0
 04 STO 00
 05♦LBL A
 06 "MAX NUM
BER?"
 07 PROMPT
 08 1
 09 +
 10 STO 04
 11♦LBL "AGN
"
 12 0
 13 STO 08
 14 STO 09
 15 10
 16 STO 07
 17 "+, -, *
, /?"
 18 AON
 19 PROMPT
 20 AOFF
 21 ASTO 06
 22♦LBL 09
 23 XEQ "RND
M"
 24 STO 02
 25 XEQ "RND
M"
 26 STO 05
 27 RCL 02
 28 XEQ IND
06
 29♦LBL "TRY
"
 30 ARCL 05
 31 ARCL 06
 32 ARCL 02
 33 "⊢=?"
 34 PROMPT
 35 RCL 03
 36 X=Y?
 37 GTO "YES
"
 38 "NO "
 39 AVIEW
 40 TONE 2
 41 TONE 2
```

Initialize.

Ask for max number.

Label to start over.

Ask which operation.

Begin loop.

Generate operands.

Generate problem.

Pose problem.

If correct, then "YES".

```
 42 FS?C 00
 43 GTO 00
 44 SF 00
 45 1
 46 ST+ 09
 47 GTO "TRY
"
 48♦LBL 00
 49 ARCL 05
 50 ARCL 06
 51 ARCL 02
 52 "⊢="
 53 ARCL 03
 54 AVIEW
 55 GTO 00
 56♦LBL "YES
"
 57 CF 00
 58 "YES"
 59 AVIEW
 60 1
 61 ST+ 08
 62♦LBL 00
 63 DSE 07
 64 GTO 09
 65 RCL 09
 66 X=0?
 67 XEQ "FF"
 68 RCL 08
 69 .1
 70 /
 71 CLA
 72 ARCL X
 73 "⊢% RIGH
T"
 74 AVIEW
 75 PSE
 76 PSE
 77 GTO "AGN
"
 78♦LBL "+"
 79 +
 80 STO 03
 81 LASTX
 82 -
 83 LASTX
 84 CLA
 85 RTN
 86♦LBL "-"
 87 -
```

If 2nd time, get new problem
else

count wrong answer and repeat problem

Display correct answer.

Display "YES".

Count right answer.

If not all problems, then repeat loop.

If no wrong answers, then play tune.

Display %RIGHT.

Start over.

Make + problem.

Make − problem.

R00 = random number
R01 = not used
R02 = n2
R03 = answer
R04 = 1 + max number
R05 = n1
R06 = kind of problem
R07 = counter
R08 = # right
R09 = # wrong

```
 88 X<=0?
 89 XEQ 00
 90 STO 03
 91 LASTX
 92 +
 93 LASTX
 94 CLA
 95 RTN
 96◆LBL 00
 97 CHS
 98 RCL 02
 99 X<> 05
100 X<> 02
101 RDN
102 RTN
103◆LBL "*"
104 *
105 STO 03
106 RCL 05
107 LASTX
108 CLA
109 RTN
110◆LBL "/"
111 X=0?
112 E↑X
113 STO 02
114 X<>Y
115 STO 03
116 *
117 STO 05
118 CLA
119 RTN
120◆LBL "RND
M"
121 RCL 00
122 9821
123 *
124 .211327
125 +
126 FRC
127 STO 00
128 SQRT
129 RCL 04
130 *
131 INT
132 RTN
133◆LBL "FF"
134 TONE 8
135 TONE 9
136 XEQ 00
```

Make ※ problem.

Make / problem.

Random number generator

Skew and scale the numbers.

Play a tune.

```
137 XEQ 00
138 TONE 8
139 TONE 8
140 TONE 8
141 TONE 7
142 TONE 8
143 TONE 8
144 TONE 7
145 TONE 8
146 TONE 9
147 XEQ 00
148 XEQ 00
149 TONE 9
150 TONE 8
151 XEQ 00
152 TONE 8
153 TONE 7
154 XEQ 00
155 TONE 7
156 TONE 6
157 RTN
158◆LBL 00
159 X<>Y
160 X<>Y
161 X<>Y
162 X<>Y
163 X<>Y
164 X<>Y
165 RTN
```

Subroutine to use up time.

Important status:
Size = 010
Fix 0
CF 29

Flags used
F00 set if wrong answer
F29 clear for no radix point

Notes

HEXADECIMAL-DECIMAL CONVERSION

This program converts numbers between the hexadecimal and decimal number systems. Decimal integers up to 1048575 and hexadecimal integers up to FFFFF can be converted by this program.

				SIZE: 021
STEP	**INSTRUCTIONS**	**INPUT**	**FUNCTION**	**DISPLAY**
1	Set status, key in the program and select USER mode.			
2	Initialize		[A]	READY
3	To convert a decimal number to hexadecimal key in the number	D	[E]	H
4	To convert a hexadecimal number to decimal key in the number in ALPHA mode	H	[E]	D
5	To convert the number back, just press E again		[E]	H or D
	NOTE: D represents an integer less than 1048576_{10} H represents an integer less than 1000000_{16}			

Example 1:

Convert 123_{10} to a hexadecimal number

Keystrokes	**Display**	**Comments**
[A]	***READY***	Initialize program
123 [E]	***7 B***	

Example 2:

Convert 123_{16} to a decimal number

Keystrokes	**Display**
123 [E]	***291.***

Programming Highlight

This program uses the digit-entry and alpha-entry flags, flags 22 and 23, to decide whether your number is in base 10 (decimal) or 16(hexadecimal). The first line of the program checks flag 22 to see if digits were input. If so, flag 23 is cleared so that the program can continue with step 6. If flag 22 is not set, flag 23 is tested, causing a branch to LBL04 if alpha data was keyed in. At the end of the program these flags are adjusted so that reconversion can be automatic.

```
01♦LBL E
02 FS?C 22
03 CF 23
04 FS? 23
05 GTO 04
06 STO 19
07 XEQ 08
08 +
09♦LBL 01
10 LASTX
11 ISG 16
12♦LBL 00
13 1 E2
14 /
15 INT
16 X≠0?
17 GTO 01
18 CLA
19 LASTX
20♦LBL 03
21 1 E2
22 *
23 ARCL IND
X
24 FRC
25 DSE 16
26 GTO 03
27 SF 23
28 ASTO X
29 BEEP
30 RTN
31♦LBL 04
32 ASTO 16
33 .00802
34 STO 17
35 0
36 STO 19
37♦LBL 05
38 0
39 STO 18
40 " "
41 ASTO Y
42 ARCL 16
43 ASTO 16
44 ASHF
45 ASTO X
46 X=Y?
47 GTO 08
48 CLA
49 ASTO Y
```

If alpha data

GTO Label 04. Convert decimal # to coded hex

Loop 1

Increment count
Dummy label to be skipped.

While digits remain, repeat loop 1.

Begin loop 3

Build up hex #.

Repeat loop 3 until R16 is 0.

Display hex #.

Set up to convert hex to decimal.

Begin loop 5.

Strip hex # apart.

If character is blank, then jump out of loop.

```
50 X=Y?
51 GTO 05
52♦LBL 06
53 RCL IND
18
54 X=Y?
55 GTO 07
56 RDN
57 ISG 18
58♦LBL 00
59 GTO 06
60♦LBL 07
61 RCL 18
62 RCL 17
63 INT
64 10↑X
65 *
66 ST+ 19
67 ISG 17
68 GTO 05
69♦LBL 08
70 16
71 STO 18
72 1
73 STO 17
74 0
75 STO 16
76 1 E2
77 STO 20
78 FS? 23
79 GTO 09
80 RCL 18
81 X<> 20
82 STO 18
83♦LBL 09
84 RCL 19
85♦LBL 10
86 RCL 20
87 /
88 STO 19
89 FRC
90 RCL 20
91 *
92 RCL 17
93 *
94 ST+ 16
95 RCL 18
96 ST* 17
97 RCL 19
98 INT
99 X≠0?
```

If character is null, then repeat loop 5.

Build coded hex #.

Count up to 5 hex characters.

Routine to store constants in proper registers and setup for conversion.

Begin loop 10. Convert number from one base to the other.

If not done,

R00 = "0"
R01 = "1"
R02 = "2"
R03 = "3"
R04 = "4"
R05 = "5"
R06 = "6"
R07 = "7"
R08 = "8"
R09 = "9"
R10 = "A"
R11 = "B"

```
100 GTO 10
101 X<> 16
102 CLA
103 FS?C 23
104 BEEP
105 RTN
106◆LBL A
107 CF 22
108 CF 23
109 "0"
110 ASTO 00
111 "1"
112 ASTO 01
113 "2"
114 ASTO 02
115 "3"
116 ASTO 03
117 "4"
118 ASTO 04
119 "5"
120 ASTO 05
121 "6"
122 ASTO 06
123 "7"
124 ASTO 07
125 "8"
126 ASTO 08
127 "9"
128 ASTO 09
129 "A"
130 ASTO 10
131 "B"
132 ASTO 11
133 "C"
134 ASTO 12
135 "D"
136 ASTO 13
137 "E"
138 ASTO 14
139 "F"
140 ASTO 15
141 "READY"
142 ASTO X
```

then repeat loop 10.

Initialization routine.

Important status:
Size =021
Fix 0

Flags used
F22 Digit entry
F23 Alpha entry

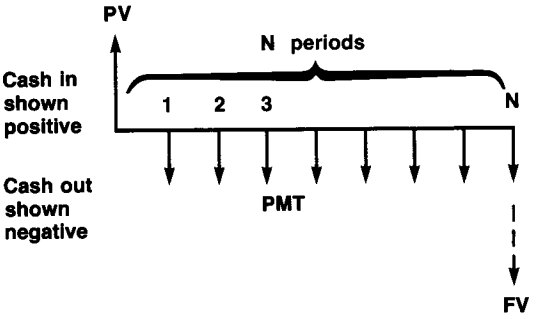
R12 = "C"
R13 = "D"
R14 = "E"
R15 = "F"
R16 = alpha

R17 = loop counter, digit counter
R18 = base constant, loop counter
R19 = decimal-coded number built here
R20 = base constant

Notes

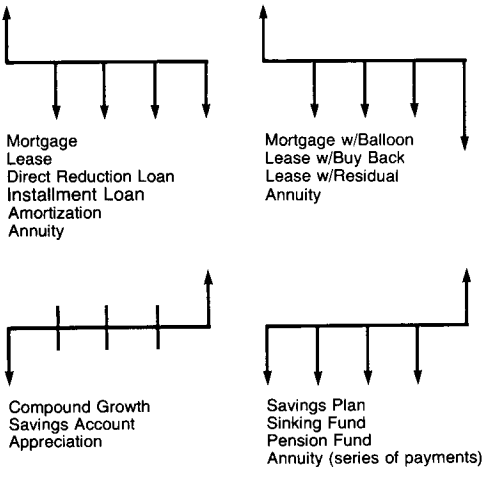
FINANCIAL CALCULATIONS

This program converts your HP-41C into a powerful financial calculator. It has the ability to solve for any of the unknowns relating to a cash flow situation as shown below.



PV = Present Value: the amount loaned, borrowed, invested, etc.
I = Periodic Interest rate.
N = Number of periods.
PMT = Payment amount: the amount paid on a loan or earned on an investment.
FV = Future Value: the amount remaining, accumulated, saved, etc.

The sketch above shows a standard loan amortization cash flow from the borrower's point of view. From the lender's point of view, PV would be shown negative and the PMT stream would be positive. By changing the signs of PV, PMT, and FV, different cash flow situtations may be realized. Cash flow diagrams for the four basic compound interest problems are presented below along with some of the more common terminology.



The five top-row keys([A] through[E])are used to enter or calculate these financial parameters. If you key in any three parameters, pressing one of the other two keys calculates the corresponding value; if you key in any four parameters, pressing the remaining key calculates its corresponding value. Previously input values can be recalled by pressing [RCL] followed by the appropriate key. The key sequence ■ [A] may be used to clear all the registers used by this program. When the registers have been cleared in this manner, the message ***N, I, PV, PMT, FV*** is put into the display to remind you of the functions of the keys.

For some combinations of values, this program fails to converge to a solution for periodic interest i. This effect may be avoided by using a different initial value for i.

Reference:

More information regarding cash-flow analysis may be found in Grant, E.L. and Ireson, W.G., *Principles of Engineering Economy*, Fourth Edition, The Ronald Press Company, New York, 1964.

				SIZE: 010
STEP	**INSTRUCTIONS**	**INPUT**	**FUNCTION**	**DISPLAY**
1	Key in the program, check status, then place the calculator in USER mode.			
2	To clear the finance registers		■[A]	N, I, PV, PMT, FV
3	Store inputs as desired			
	number of periods	N	[A]	N
	periodic interest rate, percent	I	[B]	I
	present value of investment	PV*	[C]	PV
	periodic payment	PMT*	[D]	PMT
	future value of investment	FV*	[E]	FV
4	Compute desired output			
	number of periods		[A]	N = (N)
	periodic interest rate		[B]	I = (I)% (See Note)
	present value of investment		[C]	PV = $(PV)*
	periodic payment		[D]	PMT = $(PMT)*
	future value of investment		[E]	FV = $(FV)*
5	You may return to step 4 to re-compute any of the five values or you may return to step 3 to change any or all of them.			

Note: Should the routine for i fail to return an answer, you may try your own non-zero initial value for i. For example to try a guess of 1%:

.01 [STO] 09 [XEQ] 06

*Positive for cash received, negative for cash paid out.

Example 1:

A couple purchases a $50,000 house, borrowing $40,000 at 8.5% for 30 years less one month. What is their monthly payment?

Keystrokes	Display
■ [A] 40000 [C]	***40,000.00***
8.5 [ENTER↑] 12 [÷] [B]	***0.71***
30 [ENTER↑] 12 [×] 1 [−] [A] [D]	***PMT=$-307.75***

Example 2:

The couple in example 1 sold their house 18 months later, netting $25,000. At what interest rate would they have had to invest their original $10,000 and $307.75 monthly payments to obtain $25,000?

Keystrokes	Display	
18 [A]		
25000 [E]	***25,000.00***	
10000 [CHS] [C] [B]	***I = 3.21%***	Monthly interest rate.
12 [×]	***38.51***	Annual rate

Programming Tip

This program demonstrates a technique called an "interchangeable solution." Each of the five variables in the equation can be written in terms of the remaining four. The five top-row keys are used both for storing inputs and computing outputs using the program structure outlined below.

LBL ℒ One of the labels A-J or a-e.
STO r Store the variable in register r.
FS?C22 Test the digit-entry flag and clear it.
RTN Stop here if this data was just keyed in.
} Compute the value of the unknown.
STO r Store the computed value in register r.
} Display the new value.
RTN

This building block may be repeated as many times as necessary depending on the number of variables.

```
01♦LBL A
02 STO 01
03 FS?C 22
04 RTN
05 RCL 04
06 RCL 09
07 /
08 STO 00
09 RCL 05
10 -
11 RCL 03
12 RCL 00
13 +
14 /
15 LN
16 RCL 09
17 LN1+X
18 /
19 STO 01
20 "N="
21 ARCL X
22 AVIEW
23 RTN
24♦LBL B
25 STO 02
26 1 E2
27 /
28 STO 09
29 1
30 +
31 STO 07
32 RCL 02
33 FS?C 22
34 RTN
35 RCL 04
36 X≠0?
37 GTO 01
38 RCL 05
39 RCL 03
40 /
41 CHS
42 RCL 01
43 1/X
44 Y↑X
45 1
46 -
47 STO 09
48 GTO 00
49♦LBL 01
50 RCL 05
```

Store N
If new data,
then stop,
else calculate
new N.

Display new N.

Store I and some
functions of I.

If new data,
then stop,
else
if PMT=0,
then compute
new I by
simple formula.

Else compute new I
by Newton's method.

```
51 ABS
52 RCL 04
53 RCL 01
54 *
55 RCL 03
56 +
57 ABS
58 -
59 RCL 04
60 RCL 01
61 *
62 RCL 05
63 +
64 ABS
65 RCL 03
66 ABS
67 -
68 *
69 ENTER↑
70 ABS
71 /
72 1 E-9
73 *
74 STO 09
75♦LBL 06
76 XEQ 08
77 RCL 04
78 *
79 RCL 03
80 +
81 RCL 05
82 RCL 08
83 *
84 +
85 RCL 08
86 RCL 07
87 /
88 RCL 01
89 *
90 STO 06
91 1
92 RCL 08
93 -
94 RCL 09
95 /
96 -
97 RCL 04
98 RCL 09
99 /
100 *
101 RCL 05
```

Initial guess.

Begin loop.

R00 = used
R01 = n
R02 = i
R03 = PV
R04 = PMT
R05 = FV

R06 = used
R07 = 1 + i/100
R08 = used
R09 = i/100

```
102 RCL 06
103 *
104 -
105 /
106 ST- 09
107 ABS
108 1 E-7
109 X<=Y?
110 GTO 06
111 RCL 09
112*LBL 00
113 1 E2
114 *
115 STO 02
116 "I="
117 ARCL X
118 "%"
119 AVIEW
120 RTN
121*LBL C
122 STO 03
123 FS?C 22
124 RTN
125 RCL 04
126 XEQ 08
127 *
128 RCL 05
129 RCL 08
130 *
131 +
132 CHS
133 STO 03
134 "PV=$"
135 ARCL X
136 AVIEW
137 RTN
138*LBL D
139 STO 04
140 FS?C 22
141 RTN
142 XEQ 08
143 1/X
144 RCL 03
145 RCL 05
146 RCL 08
147 *
148 +
149 *
150 CHS
151 STO 04
152 "PMT=$"
153 ARCL X
```

If ΔI not small, then repeat loop.

Display new I.

Store PV. If new data, then stop, else compute new PV.

Display new PV.

Store PMT. If new value, then stop, else compute new PMT.

Display new PMT.

```
154 AVIEW
155 RTN
156*LBL E
157 STO 05
158 FS?C 22
159 RTN
160 XEQ 08
161 RCL 04
162 *
163 RCL 03
164 +
165 RCL 08
166 /
167 CHS
168 STO 05
169 "FV=$"
170 ARCL X
171 AVIEW
172 RTN
173*LBL 08
174 1
175 XEQ 09
176 RCL 01
177 CHS
178 Y↑X
179 STO 08
180 -
181 RCL 09
182 /
183 RTN
184*LBL 09
185 RCL 09
186 1
187 +
188 STO 07
189 RTN
190*LBL a
191 CLX
192 STO 01
193 STO 02
194 STO 03
195 STO 04
196 STO 05
197 STO 09
198 "N, I, PV, PMT, F"
199 "%V"
200 AVIEW
201 RTN
```

Store FV. If new data, then stop, else compute new FV.

Subroutine to compute

$$\left(1+\frac{i}{100}\right)^{-n}$$

$$\frac{1-\left(1+\frac{i}{100}\right)^{-n}}{i/100}$$

Subroutine to compute 1 + i/100

Important status
Size = 010
Fix 2

Flags used
F22 Digit entry

Notes

ROOT FINDER

A root finder is used to find values of an independent variable, x, which cause some function f(x) of that variable to be equal to zero. These values are called the zeros of the function f(x), or the roots of the equation f(x) = 0. For example, in the equation

$$f(x) = 2x - 6$$

x = 3 is a root, because

$$f(3) = 2 \times 3 - 6 = 0$$

There are many techniques that can be employed to locate the roots of an equation. Usually root-finding algorithms (procedures) begin with an initial guess and then iterate, making better and better guesses until an acceptable solution is reached. Some algorithms fail to yield an answer (converge), iterating forever. Others, even though guaranteed to converge, require a long time.

The algorithm implemented in this program will always find a root when given initial guesses straddling an odd number of roots. If the guesses do not straddle a root properly, new ones must be chosen. Thus, the price of rapid, guaranteed convergence is that you must know certain information about your function before using this program.

Before running the root finder, it is necessary to program the function whose zeros you wish to find. This is done by pressing [GTO] [.] [.] and keying in your program. The sequence [XEQ] ROOT then begins the root finding program. It requests you to key in the name you used for your function and then prompts for the two initial guesses. If both guesses yield function values on the same side of the x-axis, the message "***F1*F2>0***" appears briefly, and you will be prompted for new guesses.

The program needs registers 01 through 07 for its own use, so register 00 and as many as are available above register 07 may be used when evaluating your function. The answer is labeled and displayed when the value of the function is less than 10^{-8}. A closer tolerance can be obtained simply by keying in a different value when the program is entered.

This program will calculate the closest obtainable approximations to a root, but may continue to iterate when the magnitude of the function evaluated at these approximations exceeds the tolerance. You can check the progress of the solution by inspecting the current guesses in registers 1 and 2 using the VIEW function. You may find it convenient to assign VIEW to some key.

References: The Illinois algorithm used here is described in M. Dowell & P. Jarratt, "A modified regula falsi method for computing the root of an equation", *BIT* 11 (1971), pp. 168-174.

A similar algorithm with slightly faster convergence was developed by the same two authors: M. Dowell & P. Jarratt, "The Pegasas method for computing the root of an equation," *BIT* 12 (1972), pp. 503-508.

SIZE: 008

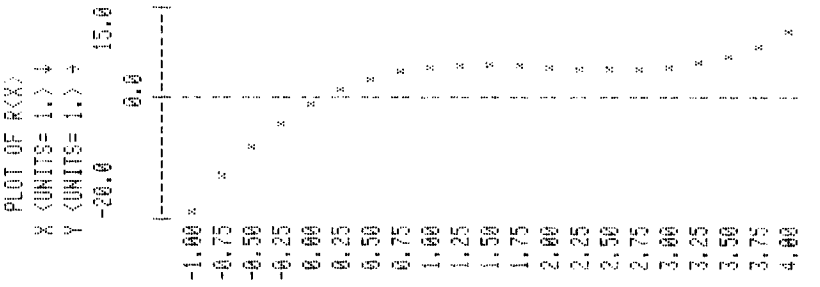
STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Set status and key in the program.			
2	Key in your function, giving it a global name (i.e., not A-J, a-e, or 00-99).			
3	Begin executing this program		XEQ ROOT	FUNCTION NAME?
4	Key in the name of your function	Name	R/S	GUESS1=?
5	Key in the first guess	X1	R/S	GUESS2=?
6	Key in the second guess and either a root will appear or, the program will return to step 5	X2	R/S	X=(ROOT) F1⁂F2>0

Example 1:

Find a value of x such that $R(x) = x^3 - 6x^2 + 11x - 1 = 0$. Note that a sketch of the function indicates a root between 0 and 1.

Keystrokes: **Display:**

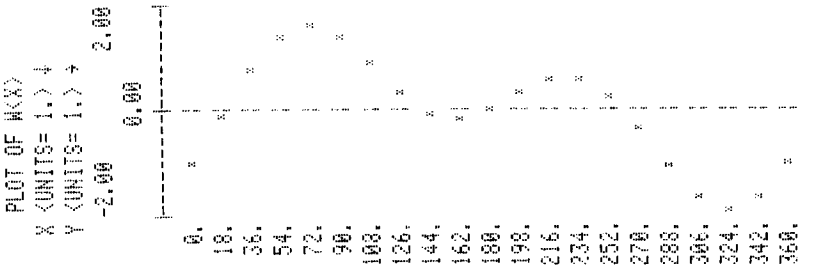
■ GTO • • PRGM
■ LBL ALPHA R ALPHA
ENTER↑ ENTER↑ ENTER↑ 6 − ×
11 + × 1 − ■ RTN
PRGM
XEQ ALPHA ROOT ALPHA — ***FUNCTION NAME?***
R R/S — ***GUESS1=?***
0 R/S — ***GUESS2=?***
1 R/S — ***X = 0.0958***



Example 2:

Find the root of $W(x) = \sin(x - 30) - \cos(2x + 60)$ which is between 200 and 300 degrees.

Keystrokes:	Display:
■ GTO • • PRGM	
■ LBL ALPHA WAVE ALPHA	
30 − SIN RCL 04	
2 × 60 + COS − ■ RTN	
PRGM	
XEQ ALPHA ROOT ALPHA	*FUNCTION NAME?*
WAVE R/S	*GUESS1=?*
200 R/S	*GUESS2=?*
300 R/S	*X = 260.0000*



Programming Highlight

The root finder program asks you to key in the name of your function. It stores that name and then executes that function indirectly as needed. Note that the function AON is executed before PROMPT so that the HP-41C will stop in ALPHA mode. The function AOFF must be executed before the next PROMPT, however, or ALPHA mode will still be on. AON and AOFF are useful for controlling the mode in which the calculator stops as a further reminder of what sort of data you should provide.

With the name of your function in register 3, the program can execute it any time with XEQ IND 03. Thus, a program which might have required modification for each function you could have wished to use, requires only the names of those functions.

FUNCTION NAME?	
AON	
PROMPT	Display the message, stopping with ALPHA mode on.
ASTO 03	The name is stored in R3.
.	
.	
.	
AOFF	Turn off ALPHA.
.	
.	
.	
XEQ IND 03	Execute the program whose name is in R3.

```
 01◆LBL "ROO
T"
 02 "FUNCTIO
N NAME?"
 03 AON
 04 PROMPT
 05 AOFF
 06 ASTO 03
 07◆LBL A
 08 "GUESS1=
?"
 09 PROMPT
 10 STO 01
 11 "GUESS2=
?"
 12 PROMPT
 13 STO 02
 14 RCL 01
 15 STO 04
 16 XEQ IND
03
 17 STO 05
 18 RCL 02
 19 STO 04
 20 XEQ IND
03
 21 STO 06
 22 RCL 05
 23 *
 24 X>0?
 25 GTO 05
 26◆LBL 00
 27 RCL 02
 28 RCL 02
 29 RCL 01
 30 -
 31 RCL 06
 32 RCL 05
 33 -
 34 /
 35 RCL 06
 36 *
 37 -
 38 STO 04
 39 XEQ IND
03
 40 STO 07
 41 X=0?
 42 GTO 04
 43 ABS
```

Ask user for the name of the function.

Store guesses.

Begin loop.

New x.

If f(x)=0 then done.

```
 44 1 E-8
 45 X>Y?
 46 GTO 04
 47 RCL 07
 48 RCL 06
 49 *
 50 X>0?
 51 GTO 01
 52 RCL 02
 53 STO 01
 54 RCL 06
 55 STO 05
 56◆LBL 02
 57 RCL 04
 58 STO 02
 59 RCL 07
 60 STO 06
 61 GTO 00
 62◆LBL 01
 63 2
 64 ST/ 05
 65 GTO 02
 66◆LBL 04
 67 "X="
 68 ARCL 04
 69 PROMPT
 70◆LBL 05
 71 "F1*F2>0
"
 72 AVIEW
 73 PSE
 74 GTO A
 75 .END.
```

Important status:
Size = 008
DEG
Fix 4

Tolerance value.
If $|f(x)| < 1E-8$ then done.

Select new guesses per requirements of Illinois algorithm.

Done.

Display answer.

Error message.

Return to input

R00 = unused
R01 = X1
R02 = X2
R03 = Name
R04 = X
R05 = f(X1)
R06 = f(X2)
R07 = f(X3)

CURVE FITTING

For a set of data points (x_i, y_i), $i = 1, 2, \ldots, n$, this program can be used to fit the data to any of the following curves:

1. Straight line (linear regression): $y = a + bx$.
2. Exponential curve: $y = ae^{bx}$ $(a > 0)$,
3. Logarithmic curve: $y = a + b \ln x$,
4. Power curve: $y = ax^b$ $(a > 0)$.

The regression coefficients a and b are found by solving the following equivalent system of linear equations.

$$An + B\Sigma X_i = \Sigma Y_i$$

$$A\Sigma X_i + B\Sigma X_i^2 = \Sigma Y_i X_i$$

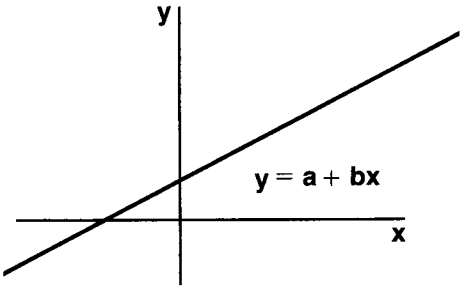
The relations of the variables are defined by the following:

Regression	A	B	X_i	Y_i
Linear	a	b	x_i	y_i
Exponential	ln a	b	x_i	$\ln y_i$
Logarithmic	a	b	$\ln x_i$	y_i
Power	ln a	b	$\ln x_i$	$\ln y_i$

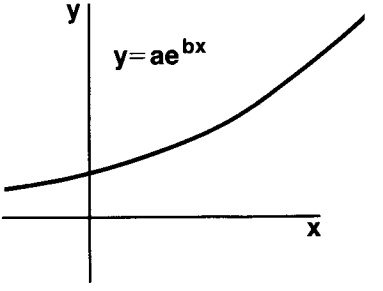
The coefficient of determination is:

$$R^2 = \frac{A\Sigma Y_i + b\Sigma X_i Y_i - \frac{1}{n}(\Sigma Y_i)^2}{\Sigma(Y_i^2) - \frac{1}{n}(\Sigma Y_i)^2}$$

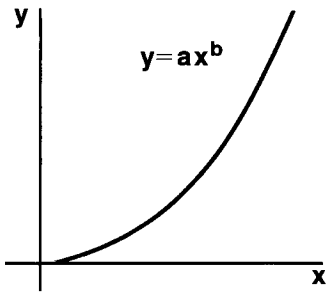
Linear Regression



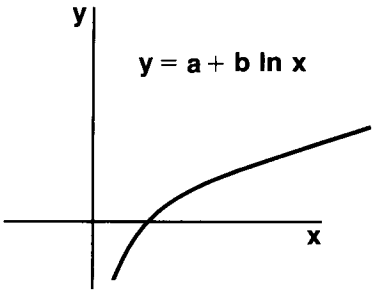
Exponential Curve Fit



Power Curve Fit



Logarithmic Curve Fit



Remarks:

1. The program applies the least square method, either to the original equations (straight line and logarithmic curve) or to the transformed equations (exponential curve and power curve).
2. Negative and zero values of x_i will cause a calculator error for logarithmic curve fits. Negative and zero values of y_i will cause a machine error for exponential curve fits. For power curve fits both x_i and y_i must be positive, non-zero values.
3. As the differences between x and/or y values become small, the accuracy of the regression coefficients will decrease.

				SIZE: 016
STEP	**INSTRUCTIONS**	**INPUT**	**FUNCTION**	**DISPLAY**
1	Set status and key in the program			
2	Initialize the program for STRAIGHT LINE or for EXPONENTIAL CURVE or for LOGARITHMIC CURVE or for POWER CURVE		 XEQ LIN XEQ EXP XEQ LOG XEQ POW	 LIN EXP LOG POW
3	Repeat step 3 and 4 for i=1,2,..., n input: x_i y_i	 x_i y_i	 ENTER↑ A	 (i)
4	If you made a mistake in inputting x_k and y_k, then correct by→	x_k y_k	ENTER↑ C	 (k−1)
5	Calculate R^2 and regression coefficients a and b		E R/S R/S	R2=(R^2) a=(a) b=(b)

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
6	Calculate estimated y from regression, input x	x	R/S	Y.=(ŷ)
7	Repeat step 6 for different x's			
8	Repeat step 5 if you want the results again			
9	To use the same program for another set of data, initialize the program by →		■ A	LIN or EXP or LOG or POW
	then go to step 3			
10	To use another program, go to step 2			

Example 1:

Fit a straight line to the following set of data and compute $\hat{y}$ for x = 37 and x = 35.

x_i	40.5	38.6	37.9	36.2	35.1	34.6
y_i	104.5	102	100	97.5	95.5	94

Keystrokes:	Display:	
XEQ ALPHA LIN ALPHA	*LIN*	
40.5 ENTER↑ 104.5 A	*1.00*	
38.6 ENTER↑ 102 A	*2.00*	
37.9 ENTER↑ 100 A	*3.00*	
36.2 ENTER↑ 97.5 A	*4.00*	
35.2 ENTER↑ 95.5 A	*5.00*	Oops!
35.2 ENTER↑ 95.5 C	*4.00*	Correct error.
35.1 ENTER↑ 95.5 A	*5.00*	Use proper values.
34.6 ENTER↑ 94 A	*6.00*	
E	*R2 = 0.99*	
R/S	*a = 33.53*	
R/S	*b = 1.76*	
37 R/S	*Y. = 98.65*	
35 R/S	*Y. = 95.13*	

Example 2:

Fit an exponential curve to the following set of data and compute $\hat{y}$ for x = 1.5 and x = 2.

x_i	.72	1.31	1.95	2.58	3.14
y_i	2.16	1.61	1.16	.85	0.5

Keystrokes:	**Display**	
XEQ ALPHA EXP ALPHA	***EXP***	
.72 ENTER↑ 2.16 A	***1.00***	
1.31 ENTER↑ 1.61 A	***2.00***	
1.95 ENTER↑ 1.16 A	***3.00***	If you don't make a mistake you can skip two steps.
2.58 ENTER↑ .85 A	***4.00***	
3.15 ENTER↑ .05 A	***5.00***	
3.15 ENTER↑ .05 C	***4.00***	
3.14 ENTER↑ 0.5 A	***5.00***	
E	***R2 = 0.98***	
R/S	***a = 3.45***	
R/S	***b = −0.58***	
1.5 R/S	***Y. = 1.44***	
2.0 R/S	***Y. = 1.08***	

Example 3:

Fit a logarithmic curve to the following set of data and compute $\hat{y}$ for x = 8 and x = 14.5.

x_i	3	4	6	10	12
y_i	1.5	9.3	23.4	45.8	60.1

Keystrokes:	**Display:**	
XEQ ALPHA LOG ALPHA	***LOG***	
3 ENTER↑ 1.5 A	***1.00***	
4 ENTER↑ 9.3 A	***2.00***	
6 ENTER↑ 23.4 A	***3.00***	
10 ENTER↑ 45.8 A	***4.00***	
12 ENTER↑ 6.01 A	***5.00***	Another mistake!
12 ENTER↑ 6.01 C	***4.00***	
12 ENTER↑ 60.1 A	***5.00***	
E	***R2 = 0.98***	
R/S	***a = −47.02***	
R/S	***b = 41.39***	
8 R/S	***Y. = 39.06***	
14.5 R/S	***Y. = 63.67***	

Example 4:

Fit a power curve to the following set of data and compute $\hat{y}$ for x = 18 and x = 23.

x_i	10	12	15	17	20	22	25	27	30	32	35
y_i	0.95	1.05	1.25	1.41	1.73	2.00	2.53	2.98	3.85	4.59	6.02

Keystrokes:	**Display:**	
XEQ ALPHA POW ALPHA	***POW***	
10 ENTER↑ 0.95 A	***1.00***	
12 ENTER↑ 1.05 A	***2.00***	
15 ENTER↑ 1.25 A	***3.00***	
17 ENTER↑ 1.41 A	***4.00***	
20 ENTER↑ 1.73 A	***5.00***	
22 ENTER↑ 2.00 A	***6.00***	
25 ENTER↑ 2.53 A	***7.00***	
27 ENTER↑ 2.98 A	***8.00***	
30 ENTER↑ 3.85 A	***9.00***	
32 ENTER↑ 4.59 A	***10.00***	
35 ENTER↑ 60.2 A	***11.00***	
35 ENTER↑ 60.2 C	***10.00***	Error correction again.
35 ENTER↑ 6.02 A	***11.00***	
E	***R2 = 0.94***	
R/S	***a = 0.03***	
R/S	***b = 1.46***	
18 R/S	***Y. = 1.76***	
23 R/S	***Y. = 2.52***	

Programming Highlight

This program uses a single section of code for most of the calculations it needs to do. Since each of the four types of curve fitting requires the input data to be in a different form, it would seem that a different program should be used for each curve type. Instead, each of the set-up programs, LIN, LOG, EXP, and POW, stores a code in register 00. Then the single function on line 32, XEQ IND 00, takes care of the four different ways of processing the input data by executing the function whose label is stored in register 00.

```
 01*LBL "LIN
"
 02 5
 03 "LIN"
 04 GTO 13
 05*LBL "EXP
"
 06 6
 07 "EXP"
 08 GTO 13
 09*LBL "LOG
"
 10 7
 11 "LOG"
 12 GTO 13
 13*LBL "POW
"
 14 8
 15 "POW"
 16*LBL 13
 17 XEQ "INI
T"
 18 STO 00
 19 ASTO 08
 20 ΣREG 10
 21 CLΣ
 22 BEEP
 23 AVIEW
 24 STOP
 25*LBL C
 26 X<>Y
 27 XEQ IND
00
 28 Σ-
 29 STOP
 30*LBL A
 31 X<>Y
 32 XEQ IND
00
 33 Σ+
 34 STOP
 35*LBL 07
 36 LN
 37 RTN
 38*LBL 08
 39 LN
 40*LBL 06
 41 X<>Y
 42 LN
 43 X<>Y
 44 RTN
```

Linear.

Exponential.

Logarithmic.

Power.

Beep, display and set Σ registers.

Correction.

Input data.

Log.

Power and exp.

```
 45*LBL E
 46 RCL 15
 47 RCL 11
 48 RCL 10
 49 RCL 10
 50 XEQ 09
 51 STO 03
 52 RCL 12
 53 RCL 11
 54 RCL 10
 55 RCL 14
 56 XEQ 09
 57 RCL 03
 58 /
 59 STO 04
 60 XEQ IND
00
 61 STO 06
 62 RCL 15
 63 RCL 14
 64 RCL 10
 65 RCL 12
 66 XEQ 09
 67 RCL 03
 68 /
 69 STO 05
 70*LBL 03
 71 RCL 04
 72 RCL 12
 73 *
 74 RCL 05
 75 RCL 14
 76 *
 77 +
 78 RCL 12
 79 X↑2
 80 RCL 15
 81 /
 82 STO 09
 83 -
 84 RCL 13
 85 RCL 09
 86 -
 87 /
 88 "R2"
 89 XEQ 88
 90 RCL 06
 91 "a"
 92 XEQ 88
 93 RCL 05
 94 "b"
```

Calculate A, b and a, b.

R00 = Index
R01 = x
R02 = y
R03 = det
R04 = A
R05 = b
R06 = a
R07 = used
R08 = LIN or EXP or LOG or POW
R09 = (Σy) 2/n

```
 95 GTO 01
 96◆LBL 06
 97◆LBL 08
 98 E↑X
 99◆LBL 05
100◆LBL 07
101 RTN
102◆LBL 09
103 *
104 STO 07
105 RDN
106 *
107 RCL 07
108 -
109 RTN
110◆LBL 00
111 "Y."
112◆LBL 01
113 "⊢="
114 ARCL X
115 AVIEW
116 FS? 55
117 STOP
118◆LBL 04
119 GTO IND
00
120◆LBL 08
121 RCL 05
122 Y↑X
123 GTO 09
124◆LBL 06
125 RCL 05
126 *
127 E↑X
128◆LBL 09
129 RCL 06
130 *
131 GTO 00
132◆LBL 07
133 LN
134◆LBL 05
135 RCL 05
136 *
137 RCL 06
138 +
139 GTO 00
140◆LBL 88
141 "⊢="
142 ARCL X
143 AVIEW
144 RTN
```

Inverse transform

Coefficient of Determination

Calculate r^2.

Input x to calculate y.

```
145◆LBL a
146 GTO IND
08
147◆LBL "INI
T"
148 CLRG
149 CF 00
150 CF 01
151 CF 02
152 SF 21
153 SF 27
154 CF 29
155 RTN
```

Re-initialize.

For initializing.

Important status
Size = 016
Σ = 10
Fix 2

Flags used
F00
F01
F02
F21
F27
F29
F55

R10 = Σx
R11 = Σx²
R12 = Σy
R13 = Σy²
R14 = Σxy
R15 = n

Notes

VECTOR OPERATIONS

This program enables you to add, subtract, multiply or divide two vectors. Before executing any of the routines, load the stack with the vector components as shown below.

Initial Stack Configuration	Resulting Display
T v_1 Z u_1 Y v_2 X u_2	U = u V = v

where the two vectors are denoted by:

$$u_1 + iv_1 \text{ and } u_2 + iv_2$$

Note that some people prefer the alternate notation of $u + vi$, $u + jv$, or $u\mathbf{i} + v\mathbf{j}$.

				SIZE: 000
STEP	**INSTRUCTIONS**	**INPUT**	**FUNCTION**	**DISPLAY**
1	Key in the program and choose a convenient display mode. You might wish to assign the routines as shown here CADD [+] CSUB [−] CMULT [×] CDIV [÷]		[ASN] CADD [+] [ASN] CSUB [−] [ASN] CMULT [×] [ASN] CDIV [÷]	
2a	Place the inputs in the operational stack Imaginary part of first vector Real part of first vector	v_1 u_1	[ENTER↑] [ENTER↑]	
2b	Imaginary part of second vector Real part of second vector	v_2 u_2	[ENTER↑]	
3	Select the desired function Vector addition Vector subtraction Vector multiplication Vector division		CADD CSUB CMULT CDIV	U = (u), V = (v) U = (u), V = (v) U = (u), V = (v) U = (u), V = (v)
4	To use this answer as part of another vector calculation, it is not necessary to re-input what was just output. Simply continue with subsequent vectors at step 2b.			

Example 1

Add 1 + i3 to 4 + i6.

Keystrokes	Display:	
■ FIX 2		Choose a convenient display.
6 ENTER↑ 4 ENTER↑ 3 ENTER↑ 1 XEQ ALPHA CADD ALPHA	*U = 5.00, V = 9.00*	Set up the vectors.

Example 2

Evaluate $s^2 + 1$ when $s = 3 + j2$

Keystrokes	Display:	
2 ENTER↑ 3 ENTER↑ 2 ENTER↑ 3 XEQ ALPHA CMULT ALPHA	*U = 5.00, V = 12.00*	
0 ENTER↑ 1 XEQ ALPHA CADD ALPHA	*U = 6.00, V = 12.00*	Add 1 + j0.

Programming Highlight

Many problems require only one number from the user, that is, you need key in only one number before executing the desired function. Vectors, however, are each described by two numbers; and two vectors must be input before the problem can be solved. Many programs can be shortened by judicious use of the stack for input data. The implementation of this program shows how short a program can become when the user is required to be careful with his input.

Notice that if the output section is replaced with LBL "UV" RTN, the four routines can be used as subroutines to any of your programs requiring vector operations. The output values u and v are returned in the X- and Y-registers respectively.

A convenient way to use this program is to assign the various routines to the [+], [−], [×], and [÷] keys for instant execution of the functions when in USER mode.

```
 01*LBL "CSU
B"
 02 CHS
 03 X<>Y
 04 CHS
 05 X<>Y
 06*LBL "CAD
D"
 07 X<>Y
 08 RDN
 09 +
 10 RDN
 11 +
 12 R↑
 13 GTO "UV"
 14*LBL "CDI
V"
 15 R-P
 16 1/X
 17 X<>Y
 18 CHS
 19 GTO 00
 20*LBL "CMU
LT"
 21 R-P
 22 X<>Y
 23*LBL 00
 24 RDN
 25 RDN
 26 R-P
 27 R↑
 28 *
 29 RDN
 30 +
 31 R↑
 32 P-R
 33*LBL "UV"
 34 "U="
 35 ARCL X
 36 "⊦,V="
 37 ARCL Y
 38 AVIEW
 39 RTN
 40 .END.
```

Important Status:
Size = 000

Subtract.

Change sign of second vector, then add.
ADD.

Divide.

Invert second vector, then multiply.

Multiply.

Display routine.

Notes

BLACKJACK

This program plays a simple version of the card game blackjack (twenty-one). The calculator deals (without replacement) from a 104-card deck, reshuffling when all but 13 cards have been dealt. The player may bet any amount; if he doesn't place a bet, the value of his previous one will be used.

The player and dealer each receive two cards, one of the dealer's cards being exposed. The player may then either draw additional cards (hit) or not draw (stand). The object of the game is to reach, but not exceed, a score of 21 points, counting 10 for face cards, 1 or 11 for aces, and the face value for the remaining cards. If a player's first two cards count 21, he has *blackjack* and immediately collects 1½ times his bet unless the dealer also has blackjack.

When hitting, a player who draws a card bringing his score over 21 is said to "bust" or "be busted" and he loses his bet. When the player stands on a score of 21 or less, the dealer must hit his own hand until his score exceeds 16. At that point the higher hand wins and the player's bank is updated. If the player and dealer should have the same score, the bet is a *stand-off* or a *push*.

Options allowed in casino-style blackjack such as splitting pairs, going down for double, and purchasing insurance are not included in this program.

You must have an HP-41C with one additional Memory Module to run this program.

SIZE: 027

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Key in program, checks status, and assign DL, HT, and S as desired. A seed ($0 \leq$ seed < 1) may be placed on R_{00}.			
2	Store your initial bank.	bank	STO 21	
3	To shuffle the deck		XEQ SH	SHUFFLING
4	Place your bet	BET	DL	I SHOW *c** YOU HAVE *1* YOU HAVE *1 2*†
5a	Hit, then repeat this step or go to 5b or,		HT	YOU HAVE *cards*
5b	Stand, and the dealer will show his hand and then hit or stand as appropriate		S	I HAVE *cards* ⋮
6	Repeat from step 4 as desired			YOUR BANK IS $ *bank*

† NOTE: If you get blackjack in step 4, the display will show BLACKJACK, and [S(TAND)] will be executed automatically.

* *c* is any card, *cards* is a string of cards—the card numbers are linked so a 10 and a 7 will look like 107.

Example:

Shuffle the deck, key in a seed of π, and play Blackjack using a $2 bet.

Keystrokes:

■ ASN ALPHA DL ALPHA Σ+
■ ASN ALPHA HT ALPHA 1/x
■ ASN ALPHA S ALPHA √x
USER
XEQ ALPHA SH ALPHA

0 STO 21
■ π STO 00
2 DL

Display:

ASN DL 11
ASN HT 12
ASN S 13

SHUFFLING
104

Only FRC (π) is used.

NOTE: The DL function was assigned to Σ+. Remember, your calculator must be in user mode or you will get Σ+.

	I SHOW 2	
	YOU HAVE 107	
S	***I HAVE 2J***	NOTE: The S function was assigned to [√x]
	I HAVE 2JK	
	BUST	
	YOUR BANK IS $2	
DL	***I SHOW 6***	
	YOU HAVE A5	
HT	***YOU HAVE A57***	
HT	***YOU HAVE A575***	
S	***I HAVE 6K***	
	I HAVE 6K8	
	BUST	
	YOUR BANK IS $4	

Program Highlight

With the 11 registers left after keying in this program, you can write a program to play blackjack using simple playing and betting schemes. The routine shown checks registers and flags used by the blackjack program to determine whether to hit or stand. If the playing program loses, it doubles its bet, eventually winning. By adding still more memory modules to your HP-41C, more complicated playing strategies may be tried.

Notice that this program requires the data memory size to be increased to 28.

```
01◆LBL "PL"
02 2                  Place new bet
03 SF 22
04◆LBL 02
05 XEQ "DL"           Deal
06◆LBL 00
07 RCL 24             check score
08 12
09 ENTER↑             Adjustment for Ace
10 10                 If no Ace
11 FS? 07             Clear adjustment
12 CLX
13 -
14 X<=Y?              If 12 ≥ score or
15 GTO 01             If blackjack
16 FC? 09             Then stand
17 GTO 01             Otherwise hit
18 XEQ "HT"
19 GTO 00
20◆LBL 01
21 FS? 09             If no blackjack
22 XEQ "S"            Then stand
23 RCL 27
24 RCL 21
25 STO 27             Save last bank
26 -
27 X<0?               If game won,
28 GTO "PL"           Place new bet.
29 X=0?               If game drawn,
30 GTO 02             Use last bet.
31 2                  If game lost,
32 ST* 22             Double the bet.
33 GTO 02
34 END
```

```
 01*LBL "CRD
"
 02 CLA
 03 ASTO 19
 04 1
 05 STO 15
 06 RCL 00
 07 9821
 08 *
 09 .211327
 10 +
 11 FRC
 12 STO 00
 13 RCL 14
 14 *
 15 INT
 16 1
 17 +
 18*LBL 02
 19 RCL IND
15
 20 X>Y?
 21 GTO 03
 22 -
 23 ISG 15
 24*LBL 99
 25 GTO 02
 26*LBL 03
 27 DSE IND
15
 28*LBL 99
 29 DSE 14
 30 12
 31 RCL 14
 32 X>Y?
 33 GTO 04
 34 XEQ "SH"
 35*LBL 04
 36 RCL 15
 37 STO 16
 38 10
 39 X<=Y?
 40 GTO 00
 41 X<>Y
 42 STO 16
 43 1
 44 X=Y?
 45 GTO A
 46 CLA
```

Routine to get a card.

Random number generator.

If only 12 cards remain, then shuffle deck.

Store card.

```
 47 ARCL Y
 48 GTO 01
 49*LBL 00
 50 STO 16
 51 CLX
 52 10
 53 X=Y?
 54 GTO "10"
 55 1
 56 +
 57 X=Y?
 58 GTO J
 59 1
 60 +
 61 X=Y?
 62 GTO "Q"
 63 "K"
 64 GTO 01
 65*LBL A
 66 "A"
 67 CF 07
 68 GTO 01
 69*LBL "Q"
 70 "Q"
 71 GTO 01
 72*LBL J
 73 "J"
 74 GTO 01
 75*LBL "10"
 76 "10"
 77*LBL 01
 78 ASTO 19
 79 RCL 16
 80 RTN
 81*LBL "SH"
 82 "SHUFFLI
NG"
 83 AVIEW
 84 1.013
 85 ENTER↑
 86 8
 87*LBL 14
 88 STO IND
Y
 89 ISG Y
 90 GTO 14
 91 104
 92 STO 14
 93 CLD
```

Store card alpha.

Subroutine to reconstruct deck.

R00 = Random number
R01 = Aces
R02 = 2's
R03 = 3's
R04 = 4's
R05 = 5's
R06 = 6's
R07 = 7's
R08 = 8's
R09 = 9's

```
 94 CF 00
 95 CF 01
 96 CF 02
 97 CF 03
 98 CF 04
 99 RTN
100◆LBL "DL"
101 CF 09          Blackjack. No ace.
102 SF 07
103 ABS
104 INT
105 FS?C 22
106 STO 22         Use old bet or store new
107 RCL 22         bet.
108 STO 20
109 SF 06
110 CLA
111 ASTO 26
112 ASTO 25
113 XEQ "CRD       Get dealer's first card.
"
114 RCL 15
115 STO 17
116 XEQ "CRD       Get dealer's second
"                  card.
117 STO 23
118 CF 08
119 FS? 07
120 SF 08          Save dealer's A-flag.
121 CLA
122 ARCL 19
123 ARCL 25
124 ASTO 25        Dealer's hand.
125 "I SHOW
"
126 ARCL 25        Display dealer's up
127 AVIEW          card. No ace.
128 SF 07
129 0
130 STO 24
131 XEQ "CRD       Get player's card.
"
132 XEQ "PH"
133 XEQ "CRD       Get player's 2nd card.
"
134 XEQ "PH"
135 RCL 24         Display player's hand.
136 10
137 FS? 07
138 CLX
139 +
140 21
141 X≠Y?
142 SF 09          If no blackjack, then set
143 FS? 09         Flag 9.
144 RTN
145 21.5
146 STO 24
147 1.5            Blackjack.
148 ST* 20
149 "BLACKJA       Go directly to
CK"                "STAND".
150 AVIEW
151◆LBL "S"
152 CF 06          Player not busted. If not
153 FS? 07         blackjack, skip to 05.
154 GTO 05
155 11
156 RCL 24
157 X>Y?
158 GTO 05
159 10
160 ST+ 24
161◆LBL 05
162 CF 07          Reinstate Dealer's
163 FS? 08         Ace-flag.
164 SF 07
165 RCL 17         Recover Dealer's
166 STO 15         hole card.
167 XEQ 04
168 XEQ "DH"       Display Dealer's hand.
169 FS? 07         If no dealer ace, skip
170 GTO 07         to LBL 07.
171 11
172 RCL 23
173 X≠Y?
174 GTO 07
175 21.5
176 STO 23
177 "I HAVE
BLACKJAC"
178 "HK"
179 AVIEW
180 GTO 07
181◆LBL 06
```

R10 = 10's
R11 = J's
R12 = Q's
R13 = K's
R14 = # cards left in deck
R15 = counter
R16 = Value of current card
R17 = Dealer's hidden card
R18 = not used
R19 = Current card in ALPHA form

```
182 XEQ "CRD
"
183 XEQ "DH"
184*LBL 07
185 FS? 06
186 GTO 09
187 FC? 09
188 GTO 08
189 RCL 23
190 17
191 X<=Y?
192 GTO 08
193 FS? 07
194 GTO 06
195 11
196 RCL 23
197 X>Y?
198 GTO 06
199 7
200 X>Y?
201 GTO 06
202 10
203 ST+ 23
204*LBL 08
205 21.5
206 RCL 23
207 X>Y?
208 XEQ "DB"
209 RCL 24
210 -
211 X=0?
212 XEQ "P"
213 X>0?
214 SF 06
215*LBL 09
216 RCL 20
217 FS? 06
218 CHS
219 ST+ 21
220 "YOUR BA
NK IS $"
221 ARCL 21
222 AVIEW
223 RTN
224*LBL "HT"
225 XEQ "CRD
"
226 XEQ "PH"
```

Dealer hits.

Dealer hit or stand? If player busted, then settle bets. If player blackjack set the blackjack. If dealer's score is above 17, then settle. If no ace, then dealer hits.

If ace and score is between 7 and 11, then dealer hits.

Add 10 for ace.

Check for dealer bust.

Check for push.

Set bust flag if player loses settle bets.

If player loses subtract payoff.

Display new bank.

Player hits.
Get a new card.

Display new hand.

```
227 RCL 24
228 21.5
229 X>Y?
230 RTN
231 "BUST"
232 AVIEW
233 GTO 05
234*LBL "DB"
235 "BUST"
236 AVIEW
237 0
238 RTN
239*LBL "PH"
240 ST+ 24
241 CLA
242 ARCL 26
243 ARCL 19
244 ASTO 26
245 "YOU HAV
E "
246 ARCL 26
247 AVIEW
248 RTN
249*LBL "DH"
250 ST+ 23
251 CLA
252 ARCL 25
253 ARCL 19
254 ASTO 25
255 "I HAVE
"
256 ARCL 25
257 AVIEW
258 RTN
259*LBL "P"
260 "A PUSH"
261 AVIEW
262 ST* 20
```

Check for bust.

Dealer bust.

Display player's hand.

Display dealer's hand.

Take care of push.

Important status
Size = 028
Fix 00
CF 29
Flag 21 Should match Flag 55

R20 = Payoff
R21 = Player's bank
R22 =
R23 = Dealer's score
R24 = Player's score
R25 = Dealer's hand
R26 = Player's hand

Flags used
F00 clear
F01 clear
F02 clear
F03 clear
F04 clear
F06 Player busted
F07 Set = no Ace Clear = Ace
F08 Set = no dealer Ace Clear = dealer Ace
F09 Set = no blackjack Clear = blackjack
F29 *Clear to suppress decimal point*
F21 Should match the printer existence flag (F55)
F22 Keyboard entry

1000 N.E. Circle Blvd., Corvallis, OR 97330

For additional sales and service information contact your local Hewlett-Packard Sales Office or call 800/547-3400. (In Oregon call 758-1010.)

00041-90366

Printed in U.S.A