Alternate Appendix A: Using the TI-89 Calculator

This document summarizes TI-89 calculation and programming operations as they relate to the text, ***Inside Your Calculator***. Even those who do not read the book should find this brief summary useful for the recommendations are perfectly general. In what follows, key labels are in bold-face type, with parentheses indicating labels above keys. For example, to access the left square bracket, the keystrokes would be indicated as **2nd** **,** **([ ) .** Although every effort is made to make this section answer all problems related to ***Inside Your Calculator***, it may be necessary to refer to your calculator *User's Guide*.

The TI-89 calculator is a powerful instrument with many features such as the calculation of derivatives and integrals, factoring expressions, solving equations and providing algebraic solutions to problems, all of which are extremely useful but are not related to this book. For that reason, they will not be described here.

General Operation

Once you turn on your calculator, movement around any screen is accomplished by the very useful cursor movement keys. These are the four blue keys marked with white arrows for up, down, left and right located in the upper right area of the calculator face immediately below the **F4** and **F5** keys. Sometimes arrows on the calculator face indicate that you should use one of these keys to access additional information. For example, if you press **APPS** and scroll down to **8: Text Editor** you will see both a down arrow and a right arrow. Following either of these with the appropriate cursor movement key will give you additional menu items.

For the purposes of this book, you should use the **MODE** key to set the various alternatives appropriate to this text. Use **ENTER** to save these values:

```
Graph.............. FUNCTION
Current Folder..... main
Display Digits..... FLOAT 10
Angle.............. DEGREE
Exponential Format  NORMAL
Complex Format..... REAL
Vector Format...... RECTANGULAR
Pretty Print....... OFF
Split Screen....... FULL
Split 1 App........ Home
Exact/Approx....... APPROXIMATE
Base............... DEC
Unit System........ SI
```

The Home screen (what you access when you turn the calculator on or when you press **HOME**) is divided into four areas. Across the top are various screen modes obtained by use of the **F** keys. The largest or "history" area is where the most recent calculations and results appear. Below that is space where calculations are entered. And finally at the bottom of the screen are some indications of the modes in which the calculator is set. If you key **3** **÷** **7** **ENTER**, for example, the lowest line of the history area would display **3/7** and across from it the

approximate result, **.4285714286**, rounded to ten digits. Over time that area will accumulate many entries; it can be cleared by pressing **F1** followed by **8: Clear home**. **CLEAR** will erase the calculating line.

Use **ENTER** to complete a calculation or enter a program line, **ESC** to cancel an entry or to leave a mode. You can also leave a mode with **2nd** **QUIT** or **HOME**.

The TI-89 allows use of variable names with more than one letter: specifically, one to eight letters or digits with the first necessarily a letter. Thus a variable could be named **the5th**. Multiplication cannot then be indicated by simple juxtaposition; the ⊠ key is needed between variables as well as numbers; thus, for example, **ab** is not equivalent to **a⊠b**; however **5a = 5⊠a**. As in algebra, parentheses also indicate multiplication as in **(7)(5) = 7(5) = 35**. To indicate negative numbers like ⁻**5**, use the (–) key, not the − key, which represents subtraction as in **a−b**. Powers are done with the ^ key: for example, **2** ^ **3** calculates 2^3 or 8. Algebraic calculation order is observed. Thus, in the expression, **a+b^3⊠c**, the product **b^3** is calculated first, then it is multiplied by **c** and the result is finally added to **a**.

One more thing about variable names: **a** and **A** represent the same variable. We will stay with lower case letters in this document and the program listings.

Internal calculations are carried to 14 digits with displays rounded to 10.

Useful Keys

STO> This important key allows you to store numbers. For example, **5** **STO>** **X** will store the value **5** in location **X**. Similarly once **m1** is defined as a matrix, **m1[1,1]** **STO>** **m2[3,2]** will store the current value in row **1**, column **1** of matrix **m1** in row **3**, column **2** of matrix **m2**.

← erases what appears at the current cursor location. Typing replaces what is at the cursor location, but see the next instruction

2nd ← **(INS)** is what is called a toggle. It changes the way the calculator operates, remaining in that form until the same keys are pressed again. In this case the calculator normally operates in replacement or type-over mode. If the cursor is set at a given key, typing another key replaces it. This toggle changes to insert mode. Now pressing a key inserts that value, moving subsequent values to the right to make room for it. You will know which mode you are working in by the way the cursor blinks. (I find insert mode to be more convenient and I recommend it.) To return to replacement mode, key **2nd** ← **(INS)** again.

CATALOG is especially useful. It is a dictionary of functions and instructions. Although there is a key for natural log, **2nd** **X**(**LN**) on this calculator, there is no key for log, the base 10 logarithm. To obtain it, use **CATALOG**, then **4** (for **L, the first letter of log**) and scroll until you come to **log(**. Press **ENTER** and it will appear in your calculation or program. Notice that you do not have to use the alpha key to reach an alphabetical part of the catalog. Other useful functions that you can access using **CATALOG** are: **abs(**, for absolute value; **ClrDraw** to clear your graphics screen; **ClrHome**; **floor(** and **int(**, and two versions of what mathematicians call the *greatest integer function* and computer scientists call the *floor function* (it rounds down to

the nearest integer — thus **floor(π) = int(π)= 3** and **floor(⁻π) = int(⁻π) = ⁻4**); and all of the useful programming instructions. Some of these functions may be obtained by other means, use of **2nd** **5** **(MATH)**, for example, but you know that they are all here.

Now we can turn to programming features.

Programming

To Begin Entering a New Program

1. From your **HOME** screen key **APPS** then select **7:Program Editor**, and finally **3:New...** This produces a screen that lists:

```
Type:          Program>
Folder:        Main>
Variable:      _
```

2. Scroll down to the box and enter a program name in it. To do so you must first key **2nd** **alpha** to change to letter mode. Then type a program name. For example, suppose you wish to enter a program named **prog1**. You would type **2nd** **alpha** **STO>** **(P)** **2** **(R)** **–** **(O)** **7** **(G)** **alpha** **1**, that last **alpha** to return to number mode.

3. Now press **ENTER** **ENTER** to reach a screen for program entry. It will appear as:

```
:prog1()
:Prgm
:
:EndPrgm
```

4. Scroll down to the blank line and begin entering your program instructions.

To Leave and Return to a Program You Have Been Editing

You can exit your program at any time using the **HOME** key. To return to a program for further editing, key **APPS** **7:Program Editor** and then choose **1:Current** if this was your most recent program or, if not, choose **2:Open...** In this last case, you must scroll down to **Variable**, use the right cursor arrow to obtain a program list and from it select your program name.

To Run a Program Entered in Your Calculator

1. In the **HOME** screen type your program name followed by **()**, for example, **prog1().**
2. Key **ENTER** and, if appropriate, follow screen instructions for data entry, each time again keying **ENTER**.

Finding Commands

It is always possible to find program commands in **CATALOG**. But you can also find control structures with **F2**. Using **F2** is preferable because it provides the appropriate **End** statement as well. Included here are:

```
1. If
2. If...Then
```

```
      1. If...Then...EndIf
      2. If...Then...Else...EndIf
   4. For...EndFor
   5. While...EndWhile
```

You must, however, use CATALOG to find such commands as **Prompt**, **Disp** and **Pause**.

Program Control Structures

Tests

Program control requires tests. For example, you may wish to do something only if the value of **x** in your program is larger than the value of **y**. These tests are accessed by 2nd 5 **(MATH)** followed by **8:Test**. Here you will find not only the mathematical relations **>, <, ≥, ≤, =** and **≠**, but also the logical connectives, **not, and** and **or**. Warning: Do not use the keyboard = key for a test.

Unless otherwise instructed, a program proceeds line by line in the order those lines were entered. Control structures (listed under F2 and CATALOG) change this order of program operation in specific ways or stop program operation. In this book we will use many of these keys.

EndIf, EndFor and **EndWhile** As you will see in the following examples, these instructions send control back to the preceding control structure. These **End**s are like right hand parentheses with the left parentheses the preceding **If, For** or **While** instruction. Important: When one loop is embedded inside another, that loop's **End** occurs first.

If <test>
<instruction> A test, for example **If X=5**, determines whether the instruction is performed. When the test is true (in this case if **X** does equal five) then the following single program line is performed. When the test is false, that line is skipped. (No **EndIf** is necessary in this case although it will not make a difference if it is included.)

```
If <test>
Then <instruction 1>
<instruction 2>
...
```
EndIf In this case you want the **If** test to govern more than one line of code. You then write those program lines between **Then** and **EndIf**.

```
If <test>
Then <instruction 1>
<instruction 2>
...
Else <instruction 3>
<instruction 4>
...
```
EndIf If the test is true, the instructions between **Then** and **Else** are performed; if it is false, those between **Else** and **EndIf** are performed.

For is a counting instruction. It applies a series of steps one at a time[1] within indicated limits. Here is an example of a **For** loop:

```
0 STO> A
For N,1,5
A+N STO> A
EndFor
```

In this example, before we enter the **For** loop **A** is set equal to **0**. There is only one instruction between the **For** and **Next** lines but there could have been many. The **For** command line establishes a variable, **N**, that will take on the values successively between the numbers that follow. In this case the successive values are **N = 1,2,3,4** and **5**. Each time one of these values of **N** is set the instruction or instructions that follow until **EndFor** are processed. Thus the **For** loop here is a short way of processing the following five steps:

```
 0+1 STO> A    'A is now 1
 1+2 STO> A    'A is now 3
 3+3 STO> A    'A is now 6
 6+4 STO> A    'A is now 10
10+5 STO> A    'A is now 15
```

At this point the program leaves the **For** loop and any following statements are processed with **A = 15**. (Note that **N = 5** as well.)

While is a loop that is processed until the test included in the **While** statement fails. In other words, the **While** instruction is saying "While this is true, do the following until you reach **EndWhile**, then repeat the test. When the test fails, jump to the line after the **EndWhile**." Here is a simple example of a **While** loop:

```
1 STO> N
While N<12
  N×2 STO> N
WEnd
```

The instruction in this loop will be processed until **N** is no longer less than **12**. Here are the lines that will be processed:

```
1×2 STO> N    'N is now 2
2×2 STO> N    'N is now 4
4×2 STO> N    'N is now 8
8×2 STO> N    'N is now 16
```

After this final step it is no longer true that **N<12**. At this point the loop is exited and any instructions following it are processed with **N = 16**.

---

[1] It is possible to modify this by including a step size. Two examples: If the instruction is **For I,1,9,2**, then **N** would jump 2 each time, performing the loop for **N = 1, 3, 5, 7 and 9; if the instruction is For I,5,3,-1, N** would reduce one each time, processing **N = 5, 4 and 3.**

Lbl allows you to set a target line for a **Goto** instruction. Include a variable name in this line as in **Lbl start**.

Goto sends program control to the designated **Lbl** line. For example, the instruction **Goto start** would send program control to **Lbl start**, even if this means jumping out of a loop. Although this and other instructions can be useful, they can create a program that does not terminate. In that case you must manually stop program operation by pressing **ON**.[2]

Input and Output in Programs

Here are the program instructions that allow you to enter and retrieve data.

Prompt If you want to have the program user enter a number **N** at some point in your program, you need only create the line **Prompt N**. (Recall that you find **Prompt** in **CATALOG**.

When the program is run, it will stop when it comes to that line and display **N?**. Type a value and press **ENTER**. The program then continues with **N** set equal to the value you typed.

Disp This symbol provides a way to present information. If you want to have the program show the values of **M** and **N**, for example, simply enter the line:[3]

```
Disp M,N
```

If this line occurs at the end of a program or if these are the only values to be displayed, nothing further is necessary. If, however, other values are to be displayed or if the program continues beyond this point, you will need to have the program stop temporarily so you can observe these values. In that case follow this line with the instruction **Pause**. When the program with the two lines

```
Disp M,N
Pause
```

runs, it will show those two values and stop. To continue, press **ENTER**.

Output lines are useful for other purposes as well. For example, you can use lines like these entered at various points in your program to print data values as an aid to debugging — programmer's lingo for finding and correcting errors.

Matrices

First, let's understand what a matrix is. It is an array of numbers arranged in rows and columns.[4] For example, you might have the array:

---

[2] Early programmers used **Goto** instructions extensively until a computer scientist named Dykstra severely criticized such use because they created programs that looked like spider webs and were very difficult to interpret and debug. Since then this control structure has been rarely used. There are, however, situations when it best serves the programmer's purpose. **Goto** is used only a few times in appropriate places in this text.

[3] Once a program has been run, you can also access any value that occurred in that program by typing its name followed by **ENTER**. You could, for example, have omitted a **Disp N** line from a program and after the program has been run type **N ENTER**.

```
2  0  1  3
1  0  6  6
1  4  9  2
```

This is a 3 by 4 matrix. Arbitrarily, we refer to matrix dimensions and matrix entries in the order horizontal row then vertical column. (Note that you always use this order row, then column.) If we have named this matrix **m1**, we can refer to individual entries. For example **m1[3,4]** is **2**, the number in the 3rd row and 4th column.

We can also enter or change entries in a matrix. For example, the program line:

5 STO> m1[1,4]

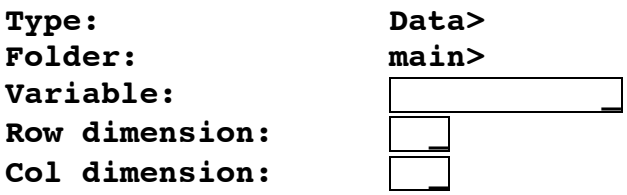
would change the **3** in that matrix example to **5**.

Preparing a Matrix for Program Use

For the few programs involving matrices used in ***Inside Your Calculator*** and in these web-based supplements, you can set up and name the necessary matrices separate from your programs. Matrices for the TI-89 allow up to 999 rows and 99 columns, but you will run out of memory if you call for too many rows and columns. Your best bet: name 4 rows and enough columns to carry data for a reasonably sized application, say 40.

To establish a matrix:

1. From your home menu key **APPS** then choose **6:Data/Matrix Editor** and then **3:New**. This will produce the screen:



2. The cursor will be flashing on **Data>**. Use the right cursor arrow to get a new screen on which you will choose **2.Matrix** by typing **2** or scrolling to choose this with **ENTER**.

3. Now scroll down to Variable. In the box enter a variable name. Although single letters are allowed, you are advised to choose a name with more than one letter. The problem is that you will be establishing a matrix with that name and you might use that same letter in a program, expecting it to represent a single value. For example, if you named your matrix simply **m**, you could not check to see if **m>5**. In the programs translated for the TI-89 I have used the names **m1**, **m2** and **m3**. The name you choose here will serve from now on as a matrix. If you choose a name already used for something else or call for too many rows and columns for your calculator memory, you will get an error message.

4. Now scroll down and enter a number of rows and columns. For the applications of this book, 5 rows and 50 columns suffice, even if you don't need them all.

---

[4] Matrices (that's the plural of matrix) are very useful mathematical structures that can be manipulated in many ways by well established rules: by addition and multiplication, for example. In this book we will be working only with single matrices and only to store and change value entries. Those matrix applications appear in Appendix L and Chapter 9.

5. When you have done this and press ENTER, you will produce a screen with part of your matrix on display. Note that all the entries will be designated **0.**

The program **BIGMULT** of Appendix L calls for a matrix, **A**, which I have renamed **m2** for the TI-89. It calls for three rows and a number of columns depending on the size of the multiplicands. The program **BIGDIV** of this website calls for a matrix, **A**, renamed **m3** for the TI-89, with five rows and a number of columns depending on the divisor, dividend and quotient. The 50 columns I have recommended will be enough to process 200 digit numbers since four digits are entered in each matrix element. It is possible, however, to work with still larger numbers by calling for more columns.

Matrix Reference

You are now ready to use a matrix like **m1** that you have defined in your program. Simply refer to matrix elements as **m1** with the number of the row and column in square brackets. For example, you could use **m1[2,3]** to refer to the element in row 2, column 3 of the matrix. These matrix elements are like other variables except that you have defined a great many of them all at once. There is another, still more important difference: you can process these variables systematically. For example, you could fill all 50 entries in the second row of **m1** with natural numbers with these three program lines:

```
For i,1,50
i STO> m1[2,i]
EndFor
```