T|MUG

T/MASTER USER'S GROUP NEWSLETTER
VOLUME 7, NUMBER 4, OCTOBER/NOVEMBER/DECEMBER 1988

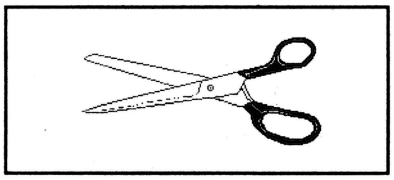
In this issue:

T/Master News

-- Peter Roizen

Price Cut

We have decided on a new marketing strategy for T/Master. It will involve more advertising on our part and more direct sales. As one ingredient to this approach, we are lowering the new-customer price of T/Master from $295 to $139. We hope to attract more users who may have already purchased a Word or DBase or Lotus. If it turns out we generate more sales than we can handle, we will adjust the price back up a bit.



This price cut will probably not be reflected in foreign countries where T/Master is distributed by others. For one, distributors have not be given a proportional discount. We simply don't have much room to maneuver at this price. Also, a great deal of the cost of software is the marketing and support behind it. Outside the U.S. these expenses are covered by your local distributors so it is reasonable for them to expect a return for their efforts.

Incidentally, if any of you know of publications that might be a good place for us to advertise we would appreciate their names and addresses. We would prefer to stay away from the classic computer magazines which are so loaded with advertisements that is impossible to get above the noise level.

Spreadsheet Noted

T/Master got a mention in an article on spreadsheets in in a recent issue of Personal Computing Magazine. Our spreadsheet was called "interesting and innovative." I think this means they did not actually try it. I know, for one, that the more I work with it, the more I feel sorry for people with cell-based spreadsheets. Have a look at "A Model for Advertising" in this issue. It's impossible to imagine a more succinct solution than T/Master's.

Christmas in Japan

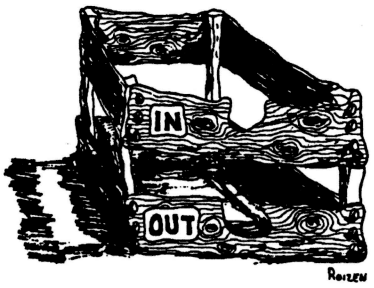
If any of you happen to be in Japan on Christmas Day tune to TOKYO Channel 6. The staff of T/Maker Research Company will be a part of a special show on the Bay Area and Silicon Valley.

If you tried to call us recently and got the answering machine message that we were too busy with TV crews, we apologize.

T/MUG Responses

It seems most of you would prefer to continue receiving T/MUG as a newsletter rather than as a diskette. So we will keep it that way. I am toying with the idea of making each issue dedicated to a particular class of problems. Perhaps T/MUG could be the manual that corresponds to a set of applications on disk. This would allow us to use T/MUG as a mailout to potential new customers as well as a service to old customers. If any of you have a good theme for a future T/MUG, let us know. I would like to give this idea at least one serious try.

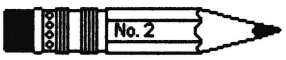
Questions and Answers



How can I add exactly one month to a date field in a database?

You can do math on dates by adding a number of days. Probably the easiest way to add a month is by adding the correct number of days. Suppose DS is the date field you start with and DE is the date field you want to end up with. The rules below should do the trick though you will need to change them when a Leap Year rolls around.

DE = DS + 31
for 4 6 9 11 end DE = DS + 30 when month (DS) = ?
DE = DS + 28 when month (DS) = 2



Can I get a "#D" substitution to put the date in a form other than "mm/dd/yy?" For my FAXes to foreign countries, I would like to show the date in the form they use.

T/Master uses one date form consistently throughout the whole program. This one form can be modified by changing some numbers in PROMPT.UTL. In your case, you want to use a different convention only on some occasions. I'm afraid all you can do for now is to build the correct date form with rules in a database. You could perhaps make this an automatic procedure that happens whenever you boot your machine. Then, instead of using "#D," you could put in a ".continue" to the file that contains the correct form of the date. (It's probably easier just to type it yourself unless it has to be part of an automated procedure.)

I want to print a graphic with some text as part of a TOP Routine. I'm am trying to use a ".SAVE A" and ".RETURN A" but it doesn't seem to work.

You are not allowed to use a ".RETURN" in a Top, Bottom, Block, etc. The only way you can accomplish side by side text and graphics within a Top Routine is to embed the text lines you want to print in the graphic itself. Read pages 14 and 15 in the Miscellaneous Booklet to see how that can be done.

Keeping an Eye on Expenses

Here are a couple of charts I make to help me keep an eye on expenses. I have a basic file in which I keep expense and income data. For present purposes, you need to know that "DDAMOUNT" is the expense amount, "EXTYPE" is the type of expense, and "TDATE" is the date of the transaction. With each year I create a new file.

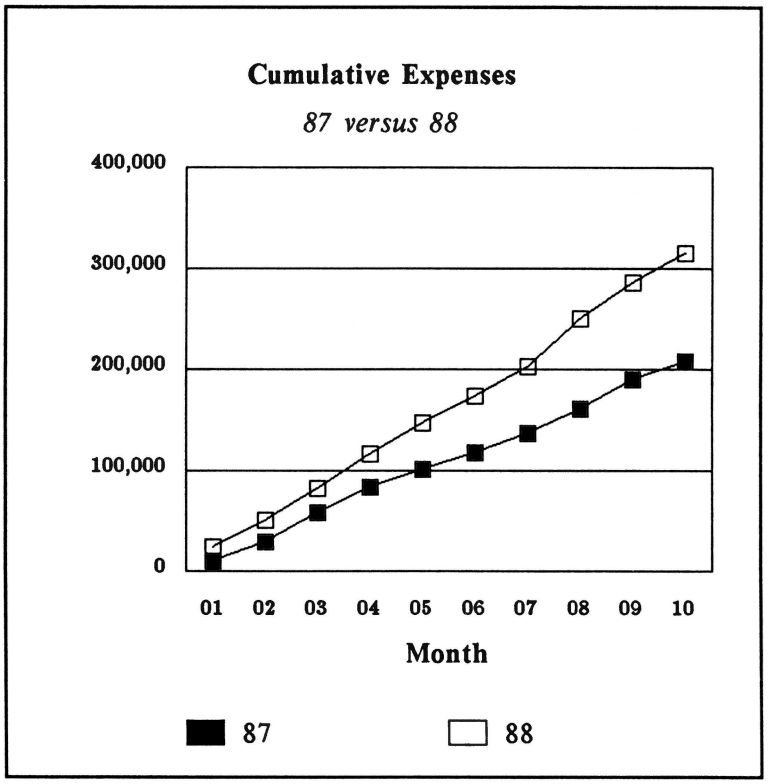
The first chart is produced by the file below:

```
select money.87 when ddamount > 0 end do
select money.88 when ddamount > 0 end do
process xxx order mon do
total drop group mon end mark junk end do
select it when ddamount1 > 0 end do
compute find CHART ex

<RECORD>
+               {TDATE#D<#8} {<mon} {DDAMOUNT>#10} {ddamount1#10} {junk      }
<END>


<rules>
mon = month (tdate)
when tdate >= date ("1/1/88")
     ddamount1 = ddamount
     ddamount = 0
<end>

CHART "87" & "88" l&p max 400000 ticks 4 key t "Cumulative Expenses"  (continued below)
      sub "87 versus 88" xlab "Month" model "999,999" tiles "1 7" c

                ------------------------------------
                     Month
                ------------------------------------
columns----------               999,999.99 999,999.99
       names                    87         88
       data                     @+a#a=     @+b#b=
                ..<HERE>
```

There's nothing very tricky here. Data for two years is SELECTed into a spreadsheet. A PROCESS command is used to pull out the month from the date and to move the data for 1988 to the second column of the spreadsheet. The data is then put in ORDER by month and aggregated with the TOTAL command. The "DROP" option is used so the result will be one line for each month.

Since the current year may not yet be complete, I next use the SELECT command to eliminate months which have not yet occurred in 1988. I find these by the fact that the expense for the month is zero--an ideal situation which never occurs.

A COMPUTE command is used to convert the two columns of data to cumulatives instead of raw figures. This takes some of the sting out of a single bad month. The result of this file is shown to the right.



More columns and equations could be added to the spreadsheet to calculate the percent change or whatever you think might be interesting. Another CHART command line could be put in the file to plot that variable.

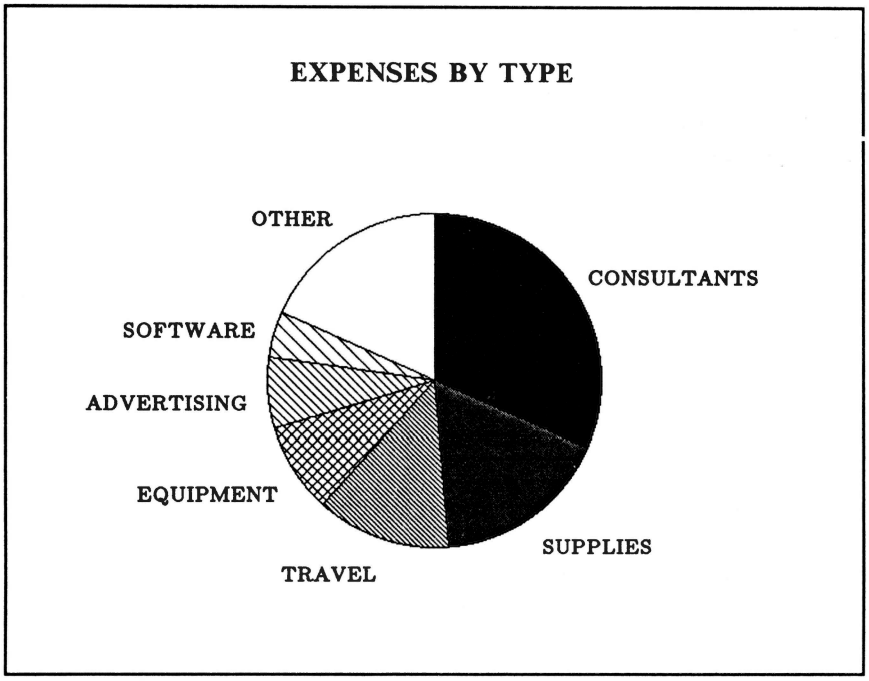
The second chart I wanted to show you is produced by the file shown on the next page. This chart, shown underneath, displays the type of expenses in pie chart format. The trick in this one is that I only want to show the six most important categories of expense and lump everything else into the category "OTHER."

After SELECTing the data, putting it in expense type ORDER, and aggregating it with the TOTAL command, I put the resulting total lines in order by decreasing amounts. Next, I use the PROCESS command to set records from the seventh record forward to have an expense type of "OTHER" and clear out the string "TOTAL" from the field name "junk." Now, the TOTAL command can again be used to aggregate all the "OTHER" records together. From there, the chart is easily produced.

Both of these examples make one use of the PROCESS command to move or manipulate data. I have some more complex reports and charts that use PROCESS a number of times on different rules to alter an emerging report. It is definitely the command to turn to when you have arrived at a given stage that is not quite what you want. Because you can attach a "WHEN" conditional to a <Rules> Definition, it is easy to arrange which rules are executed by which PROCESS command. I did not need this feature in these examples, because PROCESS is only used once in each procedure.

The nice thing about both these files is that I can alter the name of the file being SELECTed and the names of the key fields to make use of the file on another database. Thus, once I have solved the general problem of making a seven-category pie chart for one of my databases, I have effectively solved the problem for all my databases.


```
select money.88 when ddamount > 0 end do
order extype do
total group extype end drop mark junk end do
order de ddamount do
process xxx do
total group extype end drop mark junk end do
find CHART ex

<RECORD>
++               {extype<     } {DDAMOUNT>#10}  {junk      }
<END>

<rules>
junk = ""
when number >= 7
     extype = "OTHER"
<end>

CHART "AMOUNT" pie t "EXPENSES BY TYPE" c

                 Type                Amount
columns----------                 999,999.99
       names                        AMOUNT
                 ..<HERE>
```



A Model For Advertising

by Peter Roizen

In an attempt to increase direct sales of T/Master, I decided to run an advertisement. That's about as much thought as I gave to the matter. I simply waited for the next sales representative of a computer magazine to call and accepted the circumstances. I figured you get what you pay for, so figuring out how much I wanted to spend was all the figuring I did. I purchased a quarter page space in a shopper style magazine for $1,100. The advertisement rolled out of my laser printer an hour later, and I went on to another task.

Some months later, the advertisement appeared and went. The total effect was six calls for information leading to three sales. The net dollar result for the adventure was a loss of 700. At first I blamed the ridiculously high cost of advertising. Then, I blamed myself. If it wasn't my fault, how could I hope to do better the next time?

I decided to try to learn something about advertising by building a mathematical model. This model, I hoped, would let me analyze the effect an advertisement would have before I committed any money to it. One of the annoying and educative parts of building any such model is that you have to be precise. You can't say "when this is high, that is good." You need a formula and it takes thought to come up with one. At the least, making a model points out a number of things you don't know or don't understand.

Together, we will rebuild Version One of the model I built. Although it concentrates on advertising, the underlying methodology can be used on other problems. I think you will see that building a model can be quick and helpful. Gluing thoughts together is easier than you probably think.

To build this model we will use a T/Master spreadsheet. Columns will be used for cases or alternatives. The same formulae will apply to each column though the data in the columns will vary. Experience has taught me that this vertical layout of a model is the easiest to construct and maintain.

At the time, I jumped right into the task by creating a blank file for the spreadsheet. I find thinking and working on the screen of a computer is faster than going through a pencil and paper stage. I started my spreadsheet with the *columns line* below.

```
columns------------          xxx,xxx xxx,xxx xxx,xxx xxx,xxx
```

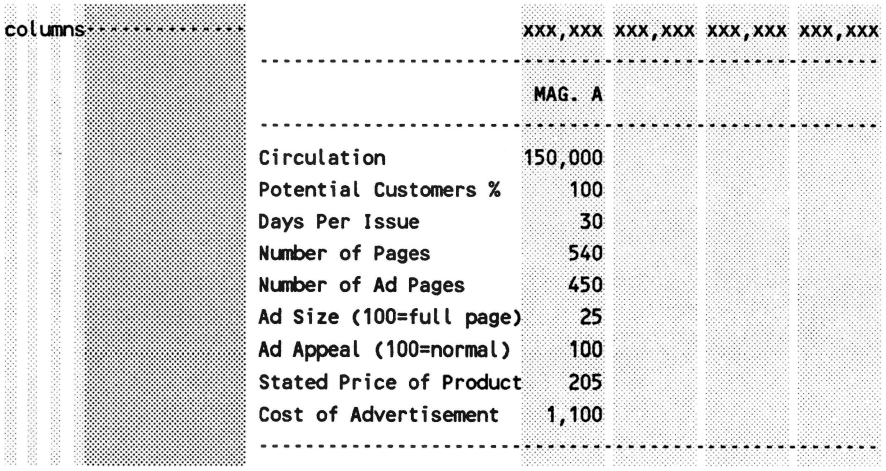
This *columns line* defines a table of four columns. Each column can have a value up to 999,999. A comma is requested separate the thousands, but no allowance has been made for a decimal point. Of course, I could change the *columns line* later if this one turned out to be inappropriate.

Next, I asked myself a question-what variables are likely to affect the number of sales I get from an advertisement? I thought out the process that occurs from the distribution of the magazine to

taking an order with questions like these:

How many people does the magazine get to?
How many of these people might be interested in our product?
How long does the magazine sit around for readers to read it?
Will they find my ad in the morass of pages and other ads?
Will they read it?
Will the price be attractive enough that they will buy it?

I entered the names of variables that would be helpful in answering these questions, a few dashed lines for decoration, and the numbers that corresponded to the experience I had had with what we will call "Magazine A". I also put in the cost of the advertisement as a variable so that I could analyze what each sale was costing in advertising dollars. At this point my spreadsheet was as shown below. I had not yet attempted to calculate anything.



columns	xxx,xxx	xxx,xxx	xxx,xxx	xxx,xxx
	MAG. A			
Circulation	150,000			
Potential Customers %	100			
Days Per Issue	30			
Number of Pages	540			
Number of Ad Pages	450			
Ad Size (100=full page)	25			
Ad Appeal (100=normal)	100			
Stated Price of Product	205			
Cost of Advertisement	1,100			

(You will notice that I scaled "Ad Size" so that a full page was 100. I did this simply to be consistent with the *columns line* I had already defined which did not allow for decimal values.)

I figured it was now time to calculate something. That something, I decided would be a value I called the "Adjusted Circulation." I intended to incorporate the first three variables or rows in my table into this calculation. At first, I thought I would multiply "Circulation" by the "Potential Customers %" by the "Days Per Issue." Clearly a magazine with twice the circulation of another (all other things equal) would be twice as useful as the other. The same logic applied to the "Potential Customers %" variable. But I did not feel the same logic could be applied to the "Days Per Issue" parameter. Would a magazine that came out every 30 days be 30 times as good as a magazine that came out daily? I didn't think so.

When you come with up a problem like this, you have to make a judgement call. So I thought to myself, how much better is a monthly magazine then a weekly magazine (all other things equal). And, how much better is a weekly magazine than a daily magazine (all other things equal). My initial feeling was that a monthly magazine was about twice as good as a weekly magazine which, itself, was about twice as good as daily magazine.

From a mathematical point of view, I needed a formula that would transform the input numbers 30, 7, and 1 into the ratios I felt were appropriate-4, 2, and 1, respectively. I didn't want to

spend hours deriving this formula since it was only a guess anyway.

A good way to decrease the differences between numbers like these is to raise them to a power of less than one (remember that raising a number to the power of 0.5 is the same as taking its square root, raising it to a power of 0.333 is taking its cube root).

I set up the little scratchpad below in T/Master to help me test what raising my numbers--30, 7, and 1--to various powers would do. T/Master interprets this series of calculations as follows: Line 1) enter ("@") 0.4 and put it in memory *b*; Line 2) enter 30 raise it to the power of the number in memory *b*, and put that result after the equal sign; Lines 3) & 4) the same as 2) but start by entering the numbers 7 and 1 instead of 30.

```
@0.4 =b
@30  ^b = 3.90
@7   ^b = 2.18
@1   ^b = 1
```

I quickly found that the value 0.4 produced approximately the kind of scaling I was after. So I decided to use it in my model to reduce the "Days Per Issue" to a transformed value that would be a suitable multiplier.

I did this in T/Master by entering the values in the row into calculators (i.e., "@"), raising those values to the 0.4 power (i.e., "^.4") and putting the resultant values into memory *d* (i.e., "=d"). The series, "@^.4=d", I wrote into the *explain zone* of the spreadsheet as shown below.

	? @^.4=d	Days Per Issue	30			

With the problem of transforming this variable behind me I went back to calculating the "Adjusted Circulation" by adding the equations below.

columns		xxx,xxx	xxx,xxx	xxx,xxx	xxx,xxx
		MAG. A			
@	Circulation	150,000			
? */100	Percent Customers %	100			
? @^.4=d	Days Per Issue	30			
	Number of Pages	540			
	Number of Ad Pages	450			
	Ad Size (100=full page)	25			
	Ad Appeal (100=normal)	100			
	Stated Price of Product	205			
	Cost of Advertisement	1,100			
? *d/3.9=	Adjusted Circulation	149,925			

Reading down the left-most position or first *lane* of the spreadsheet, let me explain what is going on. The "@" symbol enters the values for the first row. The question mark underneath tells T/Master to look in the *explain zone* for what to do. This particular notation multiplies by the "Potential Customers %" and divides by one hundred. Coming down to the last row, these values are multiplied by those in row memory *d* (the transformed values of "Days Per Issue"), divided by 3.9, and the results are reported into this row. The reason I divided by 3.9 was that I wanted the "Adjusted Circulation" of a monthly magazine with 100% potential customers to be it's regular circulation. A weekly magazine would, of course, yield an "Adjusted Circulation" lower than it's real circulation.

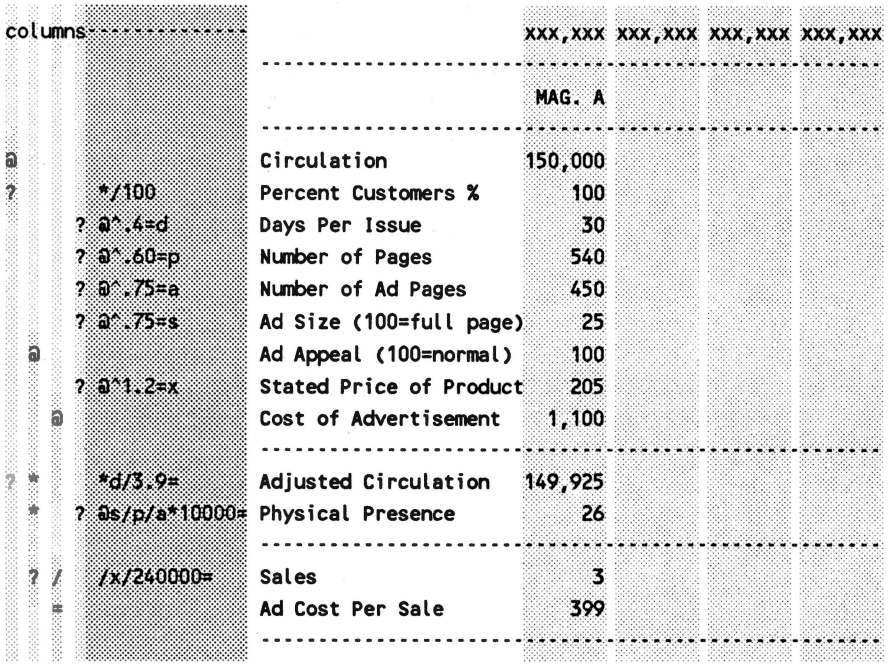
The next thing I decided to calculate was what I called the "Physical Presence" of the ad. This value would be the "Ad Size" divided by the "Number of Pages" in the magazine and the "Number of Advertising Pages" in the magazine. As with the "Days Per Issue" variable, I decided to adjust these parameters with the power operator to come up with values that made intuitive linear sense. I again made a few tests with the scratch pad feature to find numbers for the power operator that satisfied me. This led me to the spreadsheet below.

columns		xxx,xxx	xxx,xxx	xxx,xxx	xxx,xxx
		MAG. A			
@	Circulation	150,000			
? */100	Potential Customers %	100			
? @^.4=d	Days Per Issue	30			
? @^.60=p	Number of Pages	540			
? @^.75=a	Number of Ad Pages	450			
? @^.75=s	Ad Size (100=full page)	25			
	Ad Appeal (100=normal)	100			
	Stated Price of Product	205			
	Cost of Advertisement	1,100			
? *d/3.9=	Adjusted Circulation	149,925			
? @s/p/a*10000=	Physical Presence	26			

Because of the magnitude of the divisors involved in the calculation of "Physical Presence," I was forced to multiply by a large constant to get a number that would be in a suitable range.

Finally, I wanted to calculate the number of sales and the cost of advertising per sale. The formula for the number of sales was the "Ad Appeal" times the "Adjusted Circulation," times the "Physical Presence," divided by the "Stated Price." I felt the "Stated Price" had to be transformed a bit. Unlike the transformations used on other variables, I used a power greater than 1.0 which had the effect of increasing the differences between prices rather than reducing them. In other words, I believe that for products like ours, halving the price will more than double the sales. Finally, I introduced a constant so that the number of "Sales" was the number of actual sales I had achieved with the advertisement. The model thus defined is shown below with all its equations as it appeared on my screen.

The next thing I did next was to change each variable individually guessing in advance what the resulting change would be on the number of sales. By working with one variable at a time, I



columns			xxx,xxx	xxx,xxx	xxx,xxx	xxx,xxx
			MAG. A			
@		Circulation	150,000			
?	*/100	Percent Customers %	100			
	? @^.4=d	Days Per Issue	30			
	? @^.60=p	Number of Pages	540			
	? @^.75=a	Number of Ad Pages	450			
	? @^.75=s	Ad Size (100=full page)	25			
	@	Ad Appeal (100=normal)	100			
	? @^1.2=x	Stated Price of Product	205			
	@	Cost of Advertisement	1,100			
? *	*d/3.9=	Adjusted Circulation	149,925			
*	? @s/p/a*10000=	Physical Presence	26			
? /	/x/240000=	Sales	3			
=		Ad Cost Per Sale	399			

could check that I had not misentered a formula or left one variable out entirely. I could adjust the values in the formula if I felt the influence of the variable was too weak or too pronounced on the number of "Sales."

Satisfied that the model was reflective of the principles I thought I was using when building it, I tried to enter parameters for some other experiences I had had with advertising.

I had once run an advertisement in the local newspaper, say NEWSPAPER B, that had produced no sales at all. Plugging in the numbers for that experience, the model predicted a goose egg as well.

Though not exactly advertising, I had recently had a booth at a computer swap faire, say Show C. I treated the booths as pages and used the number of attendees as the circulation. The model predicted three sales--the actual number that had been made.

The numbers in the column titled "MAG. D" show where I placed the next ad I ran after having developed the model. It was a small computer newsletter (circulation 2000) in which I could purchase a full page advertisement for only $120. So far, I have had six sales from the advertisement and expect a few more. Clearly my "Add Cost Per Sale" is getting a lot better.

I have also scheduled some ads in larger magazines that the model showed would produce good results, and we are trying some direct mail. Although direct mail is expensive for the "Adjusted Circulation" you get, it's hard to beat the resulting "Physical Presence." I don't believe this model is really predictive with great accuracy, but it has provided me with a framework to evaluate the past and to make better decisions in the future. If a future experience doesn't fit the model, I will revise it by incorporating a new variable that I feel explains the deviant results. Since the first version of the model, I have added a parameter that records whether or not the magazine has a Reader Service Card. If it does, I can expect more sales, because more readers will ask for product information.

columns			xxx,xxx	xxx,xxx	xxx,xxx	xxx,xxx
			MAG. A	NWSP. B	SHOW C	MAG. D
@		Circulation	150,000	200,000	5,000	2,000
?	*/100	Percent Potential Cus.	100	7	100	100
	? @^.4=d	Days Per Issue	30	1	1	30
	? @^.60=p	Number of Pages	540	150	50	40
	? @^.75=a	Number of Ad Pages	450	75	50	10
	? @^.75=s	Ad Size (100=full page)	25	5	100	100
@		Ad Appeal (100=normal)	100	100	100	100
	? @^1.2=x	Stated Price of Product	205	295	119	205
@		Cost of Advertisement	1,100	480	50	120
? *	*d/3.9=	Adjusted Circulation	149,925	3,590	1,282	1,999
*	? @s/p/a*10000=	Physical Presence	26	65	1,608	6,148
? /	/x/240000=	Sales	3	0	3	9
=		Ad Cost Per Sale	399	4,549	18	14

I always try to keep the model in such a state that past experiences are completely explained by it. At the cost of the one or two hours it took me to develop the model, I believe I can already avoid the inefficient outcomes I experienced with Magazine A and Newspaper B. In fact, I already have a stack of a number of *media kits* for various magazines that did not stand up to the scrutiny of the model. I also have another stack of *media kits* for magazines that did.

Although this model has attacked the specific problem of where to advertise, similar models can be developed for many other purposes. I have used this general approach of listing key variables, transforming values where necessary to achieve a linear relationship, and doing some simple arithmetic for many other purposes. Like this model, most of the others fit on a single screen permitting me to tinker with their formulas and values with the greatest of ease.

For those very few of you who suggested T/Master incorporate a classic cell-based spreadsheet in some future release, you should try to make the same model with such a spreadsheet. I think you will find the effort required to build, maintain, and print the model is an order of magnitude of greater.

To Download or Not To Download

Just because you have a LaserJet and downloadable fonts doesn't mean you have to download them with the HPFONT command. When you download the font, it is fast and convenient to use but it always prints at 300 dots per inch. The CONVERT command will happily convert text using one of your font files to a bit-image picture. As a picture you could print it at a variety of different densities. You could even print the picture on another printer.

Above is a sample of University Roman (30 point) from my *Headline Typfaces 1* set that I purchased from Bitstream. I used the CONVERT command to convert the word "DOWNLOAD" and then printed it at 150 dots per inch grayed. This little trick is handy for making small signs.

Making Your First Million
(By Not Spending It!)

By Mark Nichols

When it comes down to purchasing a computer system, most people come face to face with the fact that a computer system that does more than just play games can cost a few dollars. Most people being everyone except for Donald Trump or the Federal Government, that is. And most people, when they see the amount they are going to have to shell out to get that computer system, start to rationalize the purchase by saying to themselves: "You know, I'll bet that I can use that computer to make a few bucks somehow, and get it to pay for itself." Come on, you know that you said that, so admit it. Well, this article may not show you how to make any money with your PC or T/Master, but it will certainly show you how to save your money by not spending it on the Lottery.

Here in California, the State Lottery is called Lotto 6/49. This means, for the uninitiated, that every time the game is played, 49 balls with the numbers 1 through 49 are placed in a machine that thoroughly mixes the balls and then selects 6 of the 49 numbers as the winning numbers for the current game that week. The order in which the balls are selected does not matter; what does matter is whether or not you can successfully predict in advance which six numbers are going to be selected. This may sound easy, but the odds against you are overwhelming: 1 chance in 13,983,816. And if you think that sounds terrible, it looks twice as bad when you write it out: **one chance in thirteen million, nine hundred eighty-three thousand, eight hundred sixteen.** Of course, there are other, smaller prizes in addition to the jackpot available to those incredibly lucky souls who get the 6 for 6 matchup, but when push comes to shove, it's definitely a bad bet. The T/Master files included in this article are intended to serve as a basis for the creation of a system to score multiple Lotto game tickets for each set of winning numbers and to prove to yourself that over the long run (as well as the short run, unless you are extremely lucky!), the only people to profit from the Lottery are those who run the game.

First of all, we need a file where we keep each weeks' winning numbers. Not only can we check this weeks' tickets against this file to see how much we lost (much more likely than seeing how much we won!), but those foolhardy souls who are into statistical analysis of previous weeks' winning numbers can use this file a myriad of ways as well. Following is the file that I have called "LOTTERY":

```
<record>
+ {drawdate #d}{draw1 #2 >}{draw2 #2 >}{draw3 #2 >}{draw4 #2 >}{draw5 #2 >}{draw6 #2 >}{bonus #2 >}
A {drawsum #3 >}
<end>

<scan>
Date:{drawdate#d} Numbers:{draw1 #2} {draw2 #2} {draw3 #2} {draw4 #2} {draw5 #2} {draw6 #2}
 (continuation of above line)  Bonus:{bonus #2} Sum:{drawsum #2}
<end>
```



```
<screen>
                              California Lottery
                            Winning Lotto Numbers

                          For the week of: {drawdate}

                                       1: {draw1 #2}
                                       2: {draw2 #2}
                                       3: {draw3 #2}
                                       4: {draw4 #2}
                                       5: {draw5 #2}
                                       6: {draw6 #2}
                                Bonus Number: {bonus #2}
                                             ===
                                             {drawsum #2 *}
<end>

<rules>
  drawsum = draw1 + draw2 + draw3 + draw4 + draw5 + draw6 + bonus when option = 1
<end>

<here>
```

After the **<here>** follows each weeks' winning numbers in the form of the **<record>** defined above. It's a fairly simple file, as T/Master files go; the only questionable part deals with the field called **drawsum**: some people like to do a statistical analysis of the total of each weeks' winning numbers, and after having entered the winning numbers for the current week, pressing **<F8> <1>** will add up the seven numbers (six winning numbers plus a bonus number) and put the sum in variable **drawsum**. This field can then be used for statistical reporting purposes. Some people may want to total only the six winning numbers (without the bonus number) or even to do both; the file can easily be modified to meet any particular need by adding a new variable to the record definition and modifying the rules section.

We now need a file to contain our guesses as to the winning numbers for any given week. I wrote a BASIC program to do my number selection for me (that way, if I lose, it's not my fault; I didn't pick the numbers, the computer did!); the BASIC program generates a file named THISWEEK.NUM that looks like this:

```
+  5 15 24 26 29 43
+  5  8 18 40 46 48
+ 10 23 39 41 42 43
+  1  8 20 22 23 37
```

(more lines similar to these usually follow...)

Each line in the file begins with a plus sign followed by a space as a line label, followed by six numbers from 1 to 49, each number separated by a space. The program I wrote to generate numbers also sorts the numbers in ascending order from left to right, but this is not necessary. What is necessary is to make sure that on any given line no number appears twice on the same line. Each line in the file is the equivalent of a $1 Lotto ticket; the number of lines in the file determines the number of tickets bought for each game. When you play Lotto on the computer like this, you can afford to be a big spender: I generally have my program pick $50 worth of

tickets each time I play. I picked 50 because, first, and most important, **IT'S NOT REAL MONEY!!**, and second, in theory, the overall odds for winning any prize in the Lotto 6/49 game is roughly 1 in 50; this way I figured I could usually count on coming up with one winner each game (although it seems that more often than not, I come up with The Big Zip in the way of winning tickets!). But when I really get crazy, I'll generate a few hundred tickets just to see what happens.

The file shown above with my numbers for the current game just happens to match the record format in the following file:

```
<record>
+ {my1 #2 >} {my2 #2 >} {my3 #2 >} {my4 #2 >} {my5 #2 >} {my6 #2 >} {ticknum #3 >} {score #1} {gotbonus #6 <}
- {num1 #2 >}{num2 #2 >}{num3 #2 >}{num4 #2 >}{num5 #2 >}{num6 #2 >}{bonus #2 >} }}
= {gamedate #d} }}
<end>

<globals>
{temp1}{temp2}{temp3}{temp4}{temp5}{temp6}{temp7}
<end>

<scan>
Winners:{num1 #2} {num2 #2} {num3 #2} {num4 #2} {num5 #2} {num6 #2} Bonus:{bonus #2} (Continued on next line)
   Mine:{my1 #2 >} {my2 #2 >} {my3 #2 >} {my4 #2 >} {my5 #2 >} {my6 #2 >} Score:{score #1} {gotbonus #6 <}
<end>

<screen>
           :   Ticket number: {ticknum #3}
           :     For week of: {gamedate #d}

           : Numbers selected {my1 #2}              Winning Lotto {num1 * #2}
           :     by computer: {my2 #2}              6/49 Numbers: {num2 * #2}
                              {my3 #2}                            {num3 * #2}
                              {my4 #2}                            {num4 * #2}
                              {my5 #2}                            {num5 * #2}
                              {my6 #2}                            {num6 * #2}
                                                Bonus Number: {bonus * #2}

           :Score for ticket: {score * #1} of 6 {gotbonus * #6}
<end>

<rules>
gamedate = date(option)
ticknum = number
<end>

<reference> lottery when number = 1
   when gamedate = \drawdate
      for 1 2 3 4 5 6 end temp? = \draw?
      temp7 = \bonus
<end>
```

```
<rules>
for 1 2 3 4 5 6 end num? = temp?
bonus = temp7
score = 0
for 1 2 3 4 5 6 end score = score + 1 when num? = my1
for 1 2 3 4 5 6 end score = score + 1 when num? = my2
for 1 2 3 4 5 6 end score = score + 1 when num? = my3
for 1 2 3 4 5 6 end score = score + 1 when num? = my4
for 1 2 3 4 5 6 end score = score + 1 when num? = my5
for 1 2 3 4 5 6 end score = score + 1 when num? = my6
gotbonus = "      "
for 1 2 3 4 5 6 end gotbonus = "+bonus" when bonus = my?
<end>

<here>
```

The file above, which I named SCORELOT (because it scores how well I made out in the lottery this week; pretty clever, huh?), is used in conjunction with the following command file, which just coincidentally is named SCORELOT.CMD. You will notice that SCORELOT.CMD contains a reference to another file, PROCESS.LOT, which I will explain a little later.

```
get scorelot find <here> clip a +1 insert thisweek.num save get process.lot 2 do
```

To find out how much money I lost in any given weeks' game, the first thing I have to do is update the LOTTERY file with the winning numbers for that week. This is accomplished by entering the following commands at the "What next?" prompt:

```
get lottery u ssave
```

I enter the date of the drawing, the six winning numbers, the bonus number and then press **<F8>** **<1>** to generate the total of all seven numbers. Entering an **<Esc>** **<Q>** at this point gets me out of update mode and saves the newly updated file. Now I'm ready to find out theoretically rich/poor (choose one!) I am this week. I do this by entering at the "What next?" prompt:

```
get scorelot.cmd do
```

This causes the SCORELOT file to be loaded into memory, purges out the previous weeks' records, appends file THISWEEK.NUM to the bottom of the file after the "<here>" line, saves the file, gets file PROCESS.LOT, moves line 2 in PROCESS.LOT to the top line of the screen and then performs the commands on that line. This is what file PROCESS.LOT contains:

```
                    {lastgame#d}
get scorelot process xx/xx/xx show order d score u
```

File PROCESS.LOT is written by my BASIC program and always looks exactly the same except for one piece of variable information: the date of the most recent game, replaced here with "xx/xx/xx". Not only is this date passed to the SCORELOT file by means of the PROCESS command, but the first line in PROCESS.LOT also allows me to use that date in a TRANSFER command later on. This file can easily be manually maintained by modifying the date on line 2 each week just prior to entering "get scorelot.cmd do" at the "What next?" prompt. After execution of the PROCESS command, the records in SCORELOT are then sorted in descending

order, based on field SCORE, and finally, an update command allows me to examine my results for the week. Going into scan mode shows me immediately just how many losing tickets I bought that week; anything with less than 3 matching numbers is a (theoretical) loss.

At this point, you may want to add to this system by creating a database that maintains an overall, week-to-week record of how many winning and losing tickets are bought each week; I didn't feel the need to get that sophisticated, so I did without.

There are four other files in this system that allow me to generate a single page report containing two bar charts. These charts show: 1) the total number of times each of the 49 lotto numbers has been selected throughout the history of the game (this requires that some effort be made to obtain the entire history of the lotto games' selected numbers and enter them into the LOTTERY file), and 2) a chart showing how many weeks it has been since each one of the 49 lotto numbers was selected either as one of the six winning numbers or as a bonus number. Some people may want to use these charts as an aid to selecting future ticket numbers.

The main file that is used to generate the charts/report is called LOTFREQ. Here are the contents of LOTFREQ:

```
>><<
.skip

<record>
# {row#2>}   {grtotal>#5}   {lastone>#d}    {gamessince#3>}
<end>

<table>
row 1 when draw1 = 1 or draw2 = 1 or draw3 = 1 or draw4 = 1 or draw5 = 1 or draw6 = 1 or bonus = 1
row 2 when draw1 = 2 or draw2 = 2 or draw3 = 2 or draw4 = 2 or draw5 = 2 or draw6 = 2 or bonus = 2
row 3 when draw1 = 3 or draw2 = 3 or draw3 = 3 or draw4 = 3 or draw5 = 3 or draw6 = 3 or bonus = 3
```

...and so on until we get to...

```
row 48 when draw1 = 48 or draw2 = 48 or draw3 = 48 or draw4 = 48 or draw5 = 48 or draw6 = 48 or bonus = 48
row 49 when draw1 = 49 or draw2 = 49 or draw3 = 49 or draw4 = 49 or draw5 = 49 or draw6 = 49 or bonus = 49
col grtotal 99 "0" = grtotal + 1
col lastone = drawdate when drawdate > lastone
..                          {lastgame#d}
col gamessince 999 = (date("xx/xx/xx") - lastone) / 3.5 when lastone <> "        "
<end>

<here>
```

Data to match the record definition is inserted here by TABLE command...

```
.start
.pagesize 66
.length 66
.graphic history.cht
```

<< fixed micro >>

This shows the frequency of selection of all 49 numbers since the inception of Lotto 6/49 here in California. Numbers 13 and 22 are the current leaders, having been selected 30 times during a 153 game span. Bottom chart shows number of games since each number was last selected.

```
.graphic lastpick.cht
```

File LOTFREQ contains several things: a record definition to be used by the TABLE command; a table definition which is used by the TABLE command to process the contents of the LOTTERY file; the data created by the TABLE command; dot commands that tell the PRINT command where to find the graphics files that contain the bar charts; and a paragraph separating the two charts that briefly explains their meaning. You will also notice a comment line embedded towards the end of the table definition which allows me to define a TRANSFER field (lastgame); this allows me to modify the table definition very easily after each lotto drawing by TRANSFERring the date contained in file PROCESS.LOT, rather than manually updating the table. The date is needed to calculate the number of games since each number was last selected. Lotto games are held in California twice a week; that explains (if you think about it for a while...) the formula for "col gamessince". The table is built by typing the following to the "What next?" prompt:

```
get lotstats.cmd do
```

Here are the contents of file LOTSTATS.CMD (the command line is extremely long in this file, so I have wrapped it around several lines below):

```
g lotfreq find here drop # 1 transfer process.lot table lottery save find here +1 clip b  (continues...)
        find "# 49" clip a 1 arrange 3 4 1 1 11 12 end replace # , name lotfreq.dat      (continues...)
        cd \bas asave cd \tm\tmdata g lotfreq bot -15 e save
```

This command file does the following: clears out the data lines generated by the last TABLE command; goes to the top of the file and TRANSFERs the date contained in file PROCESS.LOT; uses the table definition in the LOTFREQ file to process the data in file LOTTERY; saves the result; creates a data file used as input to my BASIC program which selects the ticket numbers and saves it to the subdirectory that contains my BASIC program; retrieves the saved LOTFREQ file, allows the user to edit the paragraph towards the bottom of the file if desired; and then saves the file one last time. After this is complete, the data in the newly created table can be used to generate the bar graph files. The last two files in this system create them; they are files MKLOTCHT.CMD and LASTPICK.CMD. Here are their contents:

MKLOTCHT.CMD

```
f here clip a 1 select lotfreq end 2 ex
CHART "Counts" seq call "C:HISTORY.CHT" max 31 minimum 13 ticks 18  (continues...)
      t "Frequency of Selection of Lotto 6/49 Numbers" (continues...)
      xlab "Lotto 6/49 Number" ylab "Times Chosen" w 8 tiles "2" c
<RECORD>
+                    {ROW#2>}   {GRTOTAL>#5}
<END>
columns                  99999
names                Num  Counts
..<here>
```

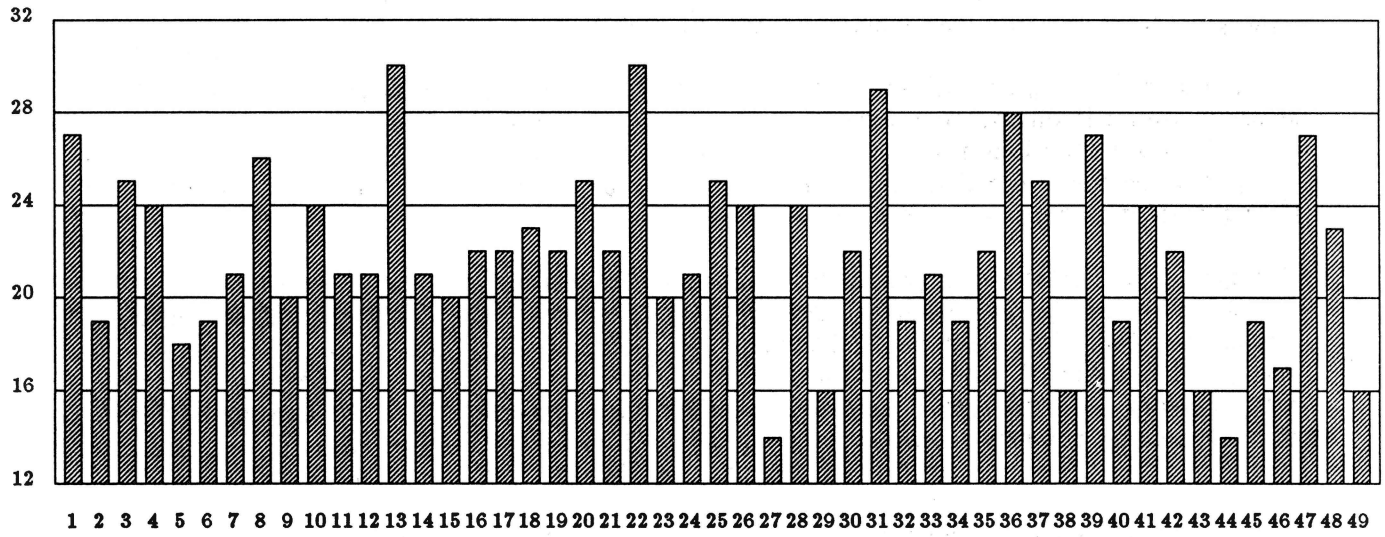

LASTPICK.CMD

```
f here clip a 1 select lotfreq end 2 ex
CHART "Counts" seq call "C:LASTPICK.CHT" max 40 ticks 20 t "Lotto Number Selection Frequency Chart" (continues...)
      xlab "Lotto 6/49 Number" ylab "Games since last picked" h 5 w 8 tiles "2" c
<RECORD>
+                    {ROW#2>}   {gamessince>#5}
<END>
columns                  99999
names                Num  Counts
..<here>
```

Neither of these two files completely does the job; the user has to examine the graphs to make sure everything looks right, possibly adjusting the minimum and maximum ranges of the graph from time to time, and issue the "save" and "end" commands manually. But once that has been done, the final result of the report looks great. Below is one of the charts included in the report.

Frequency of Selection of Lotto 6/49 Numbers

Times Chosen



Lotto 6/49 Number

I hope this has given you ideas on developing your own systems. Hopefully, this particular system will also convince you as to the hopelessness of winning big in the state lotto game without having to go broke along the way.

The Optimal Roulette Strategy

As long as the subject of gambling has come up (previous article), let's talk roulette. There is an optimal strategy for roulette. It goes like this:

1) Figure out how much you want to risk and how much you want to win.
2) Make a bet on a single number number such that if you win the bet you will have achieved your goal and can quit.
3) If you loose the bet, go back to step 2 adjusting your bet so that if you win you will recapture your loses from previous bets and win the amount you want.

For those of you unfamiliar with roulette, a bet on a single number pays 36 for 1 if it comes up. That would be fair if there were 36 numbers, but there are 38 on the wheel. The house adds a 0 and 00 to tip the odds in their favor. The strategy is optimal because it avoids two classic ways to loose--playing for ever and betting against yourself.

Every time you put a bet on the table, you can think of the house as taking a small share of it. Therefore, if your strategy involves more playing, you have less chance to win. With this strategy, once you win a single bet, you leave the table.

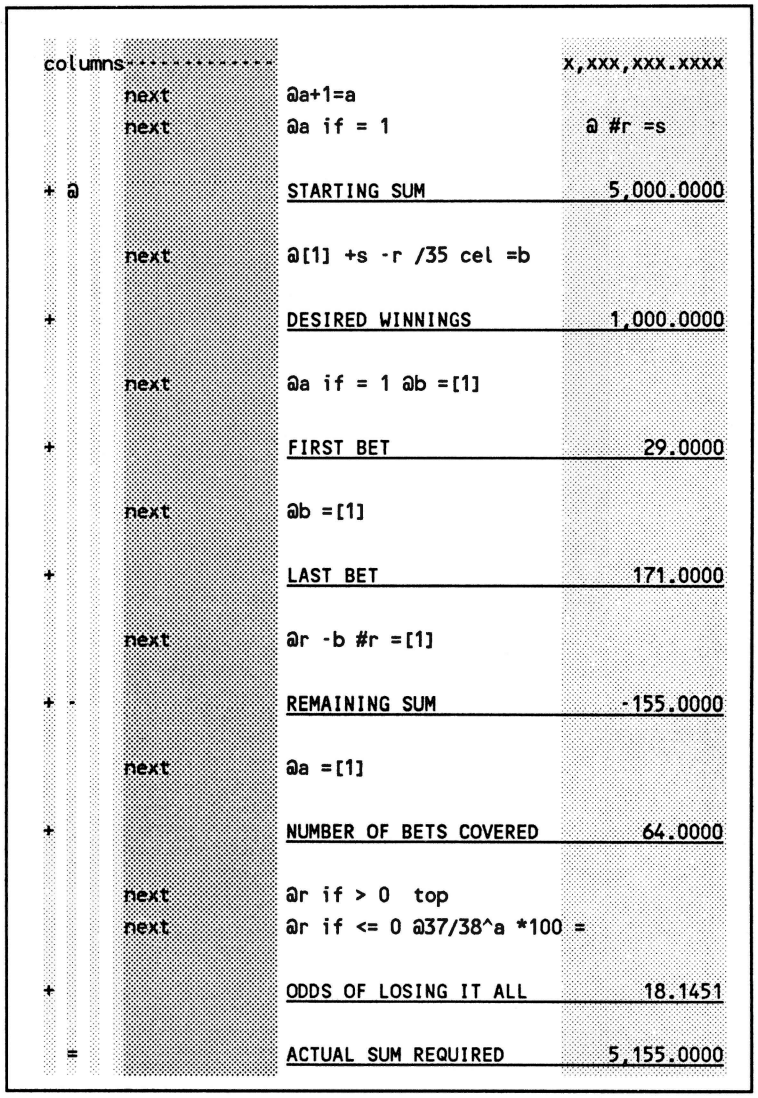
Also, you are betting on only one bet with the highest odds. If you were to bet on two numbers at once, your bet on the first number, in a sense, is against your bet on another number. You are making sure the house will win one of your bets. If you bet on all 38 numbers for example, you guarantee the house that they will get two of your chips. When you bet on red or black, it's the equivalent of betting on 18 numbers at once.

The spreadsheet on the next page calculates some numbers assuming you enter the sum you want to risk and winnings you want to achieve. The spreadsheet uses iteration. With each pass, the spreadsheet makes the next bet and assumes you loose it. The goal is to figure out how many bets you can cover. Memory *a* is used to count iterations, memory *s* is used to hold the starting sum, memory *r* is used to hold the remaining sum, and memory *b* is used to hold the bet. Because the last bet may require borrowing a few bucks from the guy next to you, the "Starting Sum" you enter is adjusted to become "Actual Sum Required" to use the strategy.

If you start, for example, with $5,155 and want to win $1,000, you have only an 18% chance of losing it all before you get to your goal. You will be able to cover 64 bets ranging from $29 to $171. Of course, you must remember that you are risking five grand to win only one.

If you start with $1,098 and want to win $1,000, you have slightly better than a fifty-fifty chance of doing so. You'll be able to cover 26 bets before you have to go home miserable. If you want to win a Porsche with $1,000, put it all on a single number. If you win, buy a used one.

ROULETTE STRATEGY SPREADSHEET



columns			x,xxx,xxx.xxxx	
	next	@a+1=a		Count interations
	next	@a if = 1	@ #r =s	Initialize memories at the first iteration
+	@	STARTING SUM	5,000.0000	
	next	@[1] +s -r /35 cel =b		Calculate the required bet
+		DESIRED WINNINGS	1,000.0000	
	next	@a if = 1 @b =[1]		Record the first bet for posterity
+		FIRST BET	29.0000	
	next	@b =[1]		Record the last bet for posterity
+		LAST BET	171.0000	
	next	@r -b #r =[1]		Figure out the remaining sum
+	-	REMAINING SUM	-155.0000	
	next	@a =[1]		Count number of bets
+		NUMBER OF BETS COVERED	64.0000	
	next	@r if > 0 top		Make next bet if money left
	next	@r if <= 0 @37/38^a *100 =		Figure losing odds if no money left
+		ODDS OF LOSING IT ALL	18.1451	
=		ACTUAL SUM REQUIRED	5,155.0000	Add initial sum to money borrowed for the last bet

Use of memories:

- *a* count of iterations and bets
- *s* starting sum
- *r* remaining sum
- *b* amount of bet

A Humongous File Problem

A few of you have asked when we will support Extended Memory. I don't yet have an answer for you, but I would like to show you how you could print up to about 10,000 labels in zip code order without much inconvenience.

For the sake of this example, suppose you have a series of files BIG.1, BIG.2, BIG.3, etc. Suppose that each of these files is crammed full with information about a company including their address. We'll assume the files are not in any particular order. For this example, we will think of the file below as a model of BIG.1, BIG.2, etc.

```
<record>
Company: {company     }
Zip:     {zip         }
<end>

<here>
Company: ACME
Zip:     94122
[etc.]
```

The classic approach to making labels would be to design the layout we want to get as a *Record Definition*, SELECT from all the various files, use the ORDER command to put things in zip code order, and then print. The problem in our example is that we won't have the space in our working file to hold 10,000 names and addresses.

Instead, we could first create the file BIG.INX which is an *index* into all the other files. The command below would do just fine. Each record in this file will use about 35 characters so we should have room for 10,000 of them.

What next? **create big.inx index big.1 zip index big.2 zip** [etc.] **order index save**

Now consider the file on the next page and the commands at the top of it. This is going to get a little tricky. If we give the DO command with this file, we will first select records from all our various files. Notice, however, that all we are selecting is the zip code. Thus, we should have room for 10,000 of them. After the selections, the file would be put in order by zip code.

Now let's consider the PROCESS command. Notice that we have some *globals* defined. These will hold values of interest and the information we need to produce the label--the company name is all we are using here. The reason we keep make these fields as globals and fields in the record is to avoid using space in our working file. The *Report definition* that produces the labels can bring in the globals just as easily as fields in the record.

Jumping down to the *Reference Definition*, the idea here is to look up the correct company name for a given zip code. In a real example, we would look up the street, city, state, etc. as well.

The reference will access for each record in the working file all records on disk that have the same index. Some of these won't necessarily have the same zip code, so we reject these right away.

Even if we find a record on disk with the same zip code, we have to be a little careful. For the first occurrence of a given zip code in our working file, we want to take the company name from the first occurrence of that zip code in our disk file. For the second occurrence of a zip code in our working file, we want to take the company name from the second occurrence of that zip code in our disk file.

Solving this problem is the reason for the fields "oldzip," "counter," and " skip." The easiest way to understand the role of the *<rules>* and the statements in the *<reference>* is to work through an example thinking what happens on a record by record basis for the records in the working file.

```
select big.1 end do
select big.2 end do
[etc.]
order zip process xxxx

<record>
+{ zip          }
<end>

<globals>
{oldzip          }
{skip          > } {counter}
{company       < }
<end>

<rules>
counter = 0
when zip <> oldzip
        skip = -1
        oldzip = zip
when zip = oldzip
        skip = skip + 1
<end>

<index1> big.inx

<reference> big.1 index1 zip when counter <= skip
when zip = \zip
        counter = counter + 1
when counter > skip
        company = \company
<end>

<report>
<page> 6
{company              }
{zip                  }
<end>

<here>
```

CP/M to OS/2

In 1979, I bought my first micro-computer, a Vector MZ. By today's standards the computer was expensive but the software was cheap. For about $300, I got CP/M and CBASIC.

CP/M, bless its soul, was described in four or five thin pamphlets. If you didn't look carefully at some of them, you might not guess that there were pages between the front and back covers. CBASIC was about half the thickness of a Playboy magazine, the print was big, and the pages were not very full. Imagine! Everything I needed to know to start my own software business could be read, studied, and reread in a few hours. That left lots of time to develop a product.

Recently, I ordered an OS/2 Development System from Microsoft for $3000. After all, I figured, if IBM endorsed it, it's bound to be the wave of the future. Now that I have received OS/2, I am beginning to wonder.

So far I have gotten seven UPS shipments incorporating 29 full-sized manuals and over 130 high-density diskettes. If that's not bad enough, a new box seems to arrive every few weeks with updates to what I already haven't had the time to read or haven't had the time to try. Yesterday I got a 13 page contract proposing to convert the support service that comes with my development kit into a "Professional Online Agreement." To be honest, I did read a few paragraphs but I can't understand it; it's all lawyers' talk. I feel like the minimum developer's environment now includes a librarian to track and index shipments and a laywer to interpret them. I'm surprised Microsoft was not thoughtful enough to include a bookcase or mobile home with the package. I sincerely wonder if they are trying to put small vendors out of business by drastically increasing their storage costs.

When I actually mustered the courage to try OS/2, I quickly found out that there was no Mouse Driver for my configuration. Considering that I have an IBM Model 80 with the letters "I-B-M" on my mouse I was a little shocked. I bought this machine precisely because I did not want to have to worry about those problems that arise from 99% compatibility. Using the keyboard instead of the mouse, I am able to hang OS/2 and force a situation where I must re-boot the system after touching only three keys. I have version 1.10 which implies to me that the other versions bombed out after a smaller number of keystrokes. At this rate, by 2155, you should be able to type a name and short address between each reboot. The wonderful graphics interface, the Presentation Manager, has me so totally confused after only a few days that I really regret having purchased this kit in the first place.

My experience has been so fruitless that my thoughts have turned to self doubt and the possibility that I might be a complete idiot. To combat this, I take out my copy of Microsoft's "Programmers At Work" book which has (among others) my picture on the cover. They wouldn't include an interview with an idiot, I think to myself, or would they?

T/Master Consultants

T/Master consultants are members of our user community who are willing to help you with your applications for a fee.

Please contact them directly for further information.

William Byers, William L. Byers Corporation, POB 223, Newcastle, Maine 04553, (207) 563-3806: Small business applications.

Dick Danielson, Strategic Advisory Consultants, Inc., P.O. Box 137, Star Prairie, WI 54026, (715) 248-3434: Fully integrated accounting system including general ledger, financial statements, accounts receivable, account analysis, operations analysis.

Robert Payne, BCS, 1210 Smith Street, Charleston, WV 25301, (304) 343-9471: Lawyer time accounting, general office accounting, A/P, A/R, payroll, general ledger.

Jeff Ribman, 344 W. 11th Street, San Pedro, CA 90731, (213) 831-0026: T/Master and T/Maker database, spreadsheet, and graphics applications for Sales/Marketing, Administration, Production and Private Practices. Networks & Hardware.

Mark Nichols, 848 Spindrift Way, San Jose, CA 95134 (408) 433-9231: My primary areas of experience include telecommunications, small business applications, report generation, billing systems and invoice generation. I have strong systems design skills and particularly enjoy helping T/Master users who may have asked themselves the question, "How can I get T/Master to do . . . ?" in the design of their own, unique applications.

Emil Widmer, CFM, Baarerstrasse 45, Postfach 708, CH-6301 Zug, Switzerland, tel:042 21 08 87: Consulting on management of uses of computers, management seminars using T/Maker, engineering and design, production engineering, personal resources, finance.

Peter Bell, Computer Factory , Box 198, Brookvale, 2100 NSW, AUSTRALIA: Any and all small business applications.

T/MUG Back Issues

If you found this issue useful, you might want to consider picking up the back issues. T/Mug has been published since 1982, each issue aiming for a mix of articles for the novice T/Maker user through the expert.

T/Mug also serves as a forum for users of particular machines and in particular industries. We try to include the applications most requested in letters and phone calls. Past issues have included: invoices, checking account management, statistics, personalized form letters, transferring data files, inventory, tracking athletic events, time calculations, and many more.

The applications, like the product, have evolved over the years, so you may find the more recent issues to be of more value. To order, please enclose this form with your check (drawn in US dollars) and mail it to the address below.

If your mailing address is in Canada, add $5 per set for shipping. If your mailing address is not in the U.S. or Canada, add $10 per set for shipping.

T/Mug back issues -- Please send me:

[ ] 1988 (two issues)	$5.00
[ ] 1987 (four issues)	$10.00
[ ] 1986 (four issues)	$10.00
[ ] 1985 (six issues)	$12.00
[ ] 1984 (six issues)	$12.00

T/MUG Subscriptions

You might also want to subscribe (or renew your subscription) to the T/MUG newsletter. This entitles you to four issues like the one in your hands.

To order, send this form with your check (drawn in US dollars) to the address below. Canadian customers add $5. Customers outside the U.S. and Canada, please add $15 for first class mailing charges.

[ ] Enter my subscription for one year -- $15.00
[ ] Renew my subscription -- $15.00 (my T/Mug mailing label is enclosed)

Name_____

Company_____

Street_____ _____

City:_____State_____Zip_____

Please make check out to T/Maker Users' Group and Mail to
T/Maker Research Company * 812 Pollard Road (#8) * Los Gatos, CA 95030

T/MUG
T/MAKER USER'S GROUP
T/MAKER Company
812 Pollard Road, Suite #8
Los Gatos, CA 95030