The "Coding Challenge" Thread

Contributors: Marc Alan Asher, Ben Bishop, Lynton Blair, Dan Baer, Alan D. Frank, Antonio Gino, Scott P. Jones, Thomas Kiendl, Stefan Moernaut, Mark Sires, Maury Pepper, Mike Pohl, Jim Self, Jim Vitek (with editing and rearranging assistance from Valerie Harvey.)

Online discourse is an important realm for exploring technical topics. Again I have selected a topic from recent M-List traffic and used a format that will treat a collection of excerpts as a contribution to *M Computing*. This approach provides print recognition to those who take the time to share their technical insights and engage in such discussion. (My ulterior motive is still: I would like to encourage eventual technical article contributions to be published in *M Computing* from some of these folks!)

For this issue, the thread topic is "coding challenge." The thread focuses on a challenge presented by Alan Frank. The opinions expressed here are not necessarily the opinions of the MTA or even of the other contributors!

Threads are conversations—they have elements of "parallel distributed processing"—parts of the conversation go on independently. It is difficult to find an appropriate "linear" arrangement for this mode of presentation and this format cannot precisely represent the dynamics of online interaction.

It is important that more M users have contact with those who are exchanging technical ideas and that the authors get credit and acknowledgement in print. The representation of the "thread" intentionally has a different appearance from the full intertwining and embedding of the Internet exchanges—each contributor is acknowledged as separately as is feasible for ideas and expression. The result is, at the same time, far less formal (given the conversational tone and cross-references between posts) than a technical article.

Alan Frank: There was a programming task that came up as I was trying to solve a word problem. I solved it the straightforward way, but along the way I thought it might be interesting to try to do it in one self-contained expression. String S contains only uppercase English letters. Some of the letters are from the upper row of the keyboard, some are from the middle, and some are from the bottom. Write an expression which will evaluate as True if and only if half the letters are from one row and the other half are from a different row, with the third row unrepresented. For example, JULY, AUGUST, and MOONBEEM would be True, while PUPPET, WEEKLY, and LUMENS would be False.

Ben Bishop: as an extrinsic:

```
ADF(S) ;returns true tvexpr under bizarre
circumstances
   NEW row1,row2,row3,lenf,lenh
   S row1="QWERTYUIOP",row2="ASDFGHJKL",
   row3="ZXCVBNM"
   S lenf=$L(S) ;full length
   S lenh=$L(S)\2 ;half length
   ;
   I lenf#2 Q 0 ;doesn't work if odd
   number of letters
   I $TR(S,row1_row2_row3)]"" Q 0
   ;not just uppercase letters
   ;
   I $TR(S,row1)=S,$L($TR(S,row2))=lenh
   Q 1 ;not row 1 & row2 has half
   I $TR(S,row2)=S,$L($TR(S,row3))=lenh
   Q 1 ;not row 2 & row3 has half
   I $TR(S,row3)=S,$L($TR(S,row1))=lenh
   Q 1 ;not row 3 & row1 has half
   ;
   Q 0 ;otherwise it does not match the
   criteria
```

Dan Baer: I try to make all my routines have only one exit point, so I'd probably change this to something like:

```
ADF(S) ;returns true tvexpr under bizarre
circumstances
  NEW row1,row2,row3,lenf,lenh,iSTATUS
  S row1="QWERTYUIOP",row2="ASDFGHJKL",
  row3="ZXCVBNM"
  S lenf=$L(S) ;full length
  S lenh=$L(S)\2 ;half length
  S iRETURN=0                ;;ADDED
  I lenf#2 S iRETURN=0
  I $TR(S,row1_row2_row3)]"" S
  iRETURN=0
  ;
  I $TR(S,row1)=S,$L($TR(S,row2))=lenh S
  iRETURN=1
  I $TR(S,row2)=S,$L($TR(S,row3))=lenh S
  iRETURN=1
  I $TR(S,row3)=S,$L($TR(S,row1))=lenh S
  iRETURN=1
  ;
  Q iRETURN
```

(The "i" in `iRETURN` tells me I'm working with a number. If it was a string, I'd use `sRETURN`.)

Antonio Gino: I just see lots of opportunity to avoid inefficient code, like the many executions that are not necessary if you exit earlier or test for certain values earlier!

Marc Alan Asher: Okay, here's my take on it:

```
FUNC(S)  ;
  S X=$TR(S,"QWERTYUIOPASDFGHJKLZXCVBNM"
  ,"11111111122222222233333333")
  Q:$TR(X,12,"")'=""&($TR(X,13,"")
  '=""&($TR(X,23)'="")!(X?1."1")!(X?1."2")
  !(X?1."3") 0
  Q $L(X,"1")=$L(X,"2")!($L(X,"1")=$L
  (X,"3"))!($L(X,"2")=$L(X,"3"))
```

Ben Bishop: Am I missing something? I believe there is a paren missing from the above. (And where the opening paren is located, it would seem that the postcond would always fail if row 2 was used, **unless** you meant:)

```
Q:$TR(X,12)]""&($TR(X,13)]"")&($TR(X,23)]"
")!(X?1."1")!(X?1."2")!(X?1."3") 0
```

Nice use of `$L(,)`! It hadn't occurred to me...

Marc Alan Asher: Oops, yes, you ARE correct. I dropped a paren there. Darn syntax checker doesn't work as well in newsgroup postings. LOL. Thanks for catching it.

Dan Baer: I somewhat agree, but one needs to take a **lot** more care to do it correctly. I feel if the quits are clear and not too much happens between them (i.e., they are clustered near the end of the extrinsic), then it is "acceptable" (but not "great").

Wouldn't it be more appropriate to use `tRETURN` since it is returning a Truth-Value?

It would seem to me that the two conditions which set `iRETURN` to 0 above should be located *after* the conditions which set it to 1 in this single-entrypoint version. I also now notice that if I used regular division (vs. integer) for setting `lenh`, then the odd-length check would not be needed.

Alan Frank: I guess I wasn't clear in my original post. I'm looking for a single expression, rather than a multi-line function definition.

Jim Self: Curiously, I have not yet seen the original question except as quoted below [in Jim's post] or a solution. Here is one.

```
$L(S)/2=$L($TR(S,"QWERTYUIOP"))&($TR(S,"QW
ERTYUIOPASDFGHJKL")="")
```

I haven't been checking c.l.m (comp.lang.mumps) lately, but on MUMPS-L, replies often show up a week or more before the message that precipitates them.

Jim Vitek: Here's my unmaintainable code version:

```
TERSE(S)  Q:S'?1.U 0 N KBD,I,R S
KBD="QWERTYUIOPASDFGHJKL
ZXCVBNM"
  F I=1:1:$L(S) S R=$F(KBD,$E(S,I))-
  2\10,R(R)=$G(R(R))+1
  Q R($O(R(""")))=R($O(R("""),-
  1))&($O(R($O(R(""")))))=$O(R("""),-1))
```

And a somewhat readable version:

```
GOOD(WORD) ;Function description:
  ; This function evaluates the input
  string, and returns true
  ; if and only if half the letters
  are from one row on a QWERTY
  ; keyboard, and the other half are
  from a different row, with
  ; the third row unrepresented.
Otherwise false is returned.
      ;Called with:
      ; WORD String of upper case alpha
      characters
      ;Returns:
      ; 1          ·'e
      ; 2        Fa⊥se
      ;
      Q:WORD'?1.U 0
      N COUNT,FIRSTROW,KBD,LASTROW,
      LETTER,NUMROWS,POS,ROW
      S KBD="QWERTYUIOPASDFGHJKLZXCVBNM"
      ;Positions 1-10 Row 1,
  11-20 Row 2
      F POS=1:1:$L(WORD) D
      . S LETTER=$E(WORD,POS)
      . S ROW=$F(KBD,LETTER)-2\10
      . S COUNT(ROW)=$G(COUNT(ROW))+1
      S ROW="" F NUMROWS=0:1 S
      ROW=$O(COUNT(ROW)) Q:ROW=""
      S FIRSTROW=$O(COUNT("")),LASTROW=$O
      COUNT(""),-1)
      Q NUMROWS=2&(COUNT(FIRSTROW)=COUNT (
      LASTROW))   ;Function
  GOOD(WORD)
```

Marc Alan Asher: Nice job on yours. I like the use of `$L`.

Scott Jones: OK - it appears that I'm so far the only one to actually meet your requirements of a single expression. I've done it in macro code so that it is more easily understood—the generated M code for the line that uses the expression is 396 characters, but that can be shortened to 386 by replacing `Quit` with `Q` and `$Select` with `$S`, so that unless an M compiler has an abysmally small line limit for routines they should be able to handle this. (Caché has a limit of 4K for each routine line-up from 510 in previous versions.)

I'm surprised nobody sent in a single expression yet though...

```
Open M for Windows NT^MAC^Full version
%RO on 13 Oct 97   9:39 AM
%spj.MAC
#; spj
#; Rather inefficient function (due to
constraint of challenge to keep as
#; a single expression) to calculate
whether a function
#;
#; Author: Scott Jones
#; Date:   October 13, 1997
#; Copyright (c) 1997 by Gandalf Software,
Inc.
#;
#define N1(%s)
$L($TR(%s,"QWERYUIOP","1111111111"),"1")
#define N2(%s)
$L($TR(%s,"ASDGHJKL","111111111"),"1")
#define N3(%s)
$L($TR(%s,"ZXCVBM","1111111"),"1")
#define
EXP(%1,%2,%3)%1=1&(%2=%3):1,%2=1&(%1=%3):1
,%3=1&(%1=%2):1
#define
SPJ(%s)$Select(%s'?1.U:0,$$$EXP($$$N1(%s),
$$$N2(%s),$$$N3(%s)),1:0)
#;
spj(s) Quit $$$SPJ(s)
#;
#; More efficient version that uses three
local variables and not just
#; a single expression
new(s) Quit:s'?1.U 0
  New n1,n2,n3 Set
  n1=$$$N1(s),n2=$$$N2(s),n3=$$$N3(s)
  Quit $Select($$$EXP(n1,n2,n3),1:0)
```

Mike Pohl: Here is another approach, although a bit soupy.

```
WRDCHK(S)   N A,B,C,L,X,Y,Z S
(A,B,C)=0,X="QWERTYUIOP",Y="ASDFGHJKL",Z="
ZXCVBNM"
  F L=1:1:$L(S) X "S
"_$S(X[ $E(S,L):"A=A",Y[ $E(S,L):"B=B",Z[ $E(
S,L):"C=C")_"+1"
Q
$S(S'?2.U:0,$L(S)#2:0,A*B*C:0,'A&(B'=C):0,
'B&(A'=C):0,'C&(A'=B):0,1:1)
```

Thomas Kiendl: What about this single expression:

```
$S(S'?1.U:0,1:$L(S)/2=$L($TR(S,$S(S=$TR(S,
"ZXCVBNM"):"ASDFGHJKL",1:"ZXCVBNM"))))
```

Jim Self: Counter examples: BEAM BRANCH
This incorrectly accepts any string with half the characters from row 3.

Scott Jones: Ah—for some reason (I'm on the mailing list - I don't read the newsgroup directly), I haven't ever gotten the original challenge (I finally saw it quoted later), and I haven't seen Maury Pepper's or your posts of solutions. (The only message I've gotten from you (via the list) just had some counter-examples, BEAM and BRANCH, for somebody else's proposed solution).

Stefan Moernaut: I didn't see the original question, but from what I understand it should have been something like half of the keys in the string should be from the first line of keys on your keyboard and the other half either from row 2 or row 3 right ? Well, I would do it like this...

```
STRTEST(S) ;
   S X="QWERTYUIOP",Y="ASDFGHJKL",
   Z="ZXCVBNM",L=$L(S)
   Q
($L($TR(S,$TR(S,X)))=(L/2))&(($TR(S,X_Y)="
")!($TR(S,X_Z)=""))
   ;
```

Maury Pepper:

```
$TR(S,"QWERTYUIOPASDFGHJKLZXCVBNM",
"1111111111AAAAAAAAA.......")?@("1("_$L(S
)_"NA,"_$L(S)_"NP,"_$L(S)_"AP)")&($L($TR(S
,"ZXCVBNM"))*2=$L(S)!($L($TR(S,"AS
DFGHJKL"))*2=$L(S)))
```

Jim Self: Not true. Maury Pepper gave a 172 character solution that checks out with 20 examples following and I have submitted two shorter solutions that also check out. Alan mentioned that Ben Bishop had a solution, but I haven't seen that yet.

Scott Jones: Ah—but the challenge was to come up with an **expression**, not a line! So far, it looks like I'm the only one to come up with an expression that works.

Jim Self: I don't think your macro code should count as a solution until it is translated to MUMPS. It has a symmetry that suggests it **might** work, but I would prefer to leave the validation to someone, such as yourself, who has a macro translator.

Maury's solution:

```
$TR(S,"QWERTYUIOPASDFGHJKLZXCVB-
NM","1111111111AAAAAAAAA.......")?@("1("_$
L(S>)_"NA,"_$L(S)_"NP,"_$L(S)_"AP)")&($L($
TR(S,"ZXCVBNM"))*2=$L(S)!($L($TR(S,"A
S>DFGHJKL"))*2=$L(S)))
```

Scott Jones: Well, I used the macro code partly to show a point—I think it is MUCH more readable, and was easier to come up with a working solution (even from me, who is NOT a Mumpster even though I've been implementing the language for almost 12 years...). Using the macro preprocessor is a huge productivity enhancer. You can simply cut and paste to get the Mumps code—anything `$$$name(args)` gets expanded.... When I get back to Massachusetts I'll post the translated code though to keep you happy.

I like the hack in this [Maury's] solution of translating to different classes of characters for pattern match that can be combined—something only a real MUMPSTER would dream up—but, performance wise, the use of the indirect pattern match...

Jim Self: My first solution was to a variant interpretation of the problem. My solution 2:

```
($TR(S,"QWERTYUIOP")=S&($L(S)/2=$L($TR(S,"
ASDFGHJKL"))))!
($TR(S,"ASDFGHJKL")=S&($L(S)/2=$L($TR(S,"Z
XCVBNM"))))!
($TR(S,"ZXCVBNM")=S&($L(S)/2=$L($TR(S,"QWE
RTYUIOP"))))
```

Scott Jones: This suffers (performance-wise) from the lack of conditional `AND` and conditional `OR` in the MUMPS language - which you fix in your next solution.

Jim Self: My solution 3:

```
$S($TR(S,"ZXCVBNM")=S:$L(S)/2=$L($TR(S,"AS
DFGHJKL")),$L(S)/2=$L($TR(S,"ZXCVB
NM")):$TR(S,"ASDFGHJKL")=S!($TR(S,"QWER-
TYUIOP")=S),1:0)
```

Scott Jones: Does this correctly return 0 for strings that are not all uppercase letters? If not, that is easy enough to fix, as I did, by adding `S'?1.U:0` at the head of the `$Select`.

So far, I like your solution # 3 the best of all that I've seen so far (including my own but then again, as it seems I'm not seeing a lot of the stuff posted on c.l.m due to the vagaries of receiving it as e-mail..., there may be some better ones floating out there...)

Maury Pepper: I've checked for `$L(S)` up to 20,000 and you're still ok. Are you confident that `$L(S)**3` will yield integer results for all standard MUMPS and all acceptable lengths of `S`?

Jim Self: Maury, What exactly did you check up to `$L(S)=20,000`? How long did it take? Your test is evidently much faster than mine. I tested for exceptions up to about `$L(S)=11,000` in about 20 hours.

Maury Pepper: I ran it on an Alpha 8200 using VAX BASIC with double precision. It took about 5 hours. Our algorithms look almost identical.

Jim Self: Here is my `Xecutable` string (broken for email) to test for exceptions to Alan's solution.

```
s quit=0
f L=2:2
q:quit
s al2=L*L*L/4
f x=L:-1:L+2\3
f y=$s(L-x>x:x,1:L-x):-1:L-x+1\2
s z=L-x-y,al1=L-x*(L-y)*(L-z)
if al1=al2
w !,$j(L,8),$j(x,8),$j(y,8),$j(z,8),$j
(al1,20)
if x-y!z w *7 s quit=1 q

L = $L(S)
x = number of characters from the row
with the most characters in S
y = number of characters from the row
with the second most characters in S
z = number of characters from the row
with the fewest characters in S
al1 = equivalent to left side of Alan's
expression
al2 = equivalent to right side of Alan's
expression
```

Maury Pepper: Jim, two points: I think the 3rd param of the 3rd `FOR` needs the same `$S(..)` treatment as the 1st param. (e.g., If L=100 & x=40, then y will start at 40 and end at 30.) I think it should end at 20. Second, to really be a true test, I think `L*L*L` needs to be `L**3` simply because I don't trust M to always return an accurate value when exponentiation is used. (BTW: like yours, my test used `L*L*L`.) That is, if Alan's test accepts a string it shouldn't, then it could be due to a true exception, or it could be due to bad M arithmetic.

From looking at your routine, I see a problem with mine which I'll repair and run again. Also, thanks to you, it should run much faster, due to the obvious indexing of L by 2, since an odd number cubed can never be divisible by 4.

Mark Sires: I haven't seen all the answers yet, and I haven't seen the original message with the challenge, but here is another:

```
(expr = phrase)
s
expr=$tr(expr,"123","444"),expr=$tr(expr,"
QWERTYUIOPASDFGHJKLZXCVBNM,"
111111111122222222233333333",expr=(expr["1"
)+(expr["2")+(expr["3" )  q
(expr=2)
```

This seems to work, and since I'm not clear if any keys other than the uppercase alpha's were to be considered or not, I ignored them. If they were to be excluded it would be simple to change the first translate to include all other excluded characters set to 4 and add `+(expr["4")` to the function after the second translate.

Lynton Blair: My solution to the problem as stated is: it's untestible !! String `S` is defined to have

some letters from each of the three rows. Therefore, no S will have letters from only two rows. (Sorry folks, the prize is mine !!)

Jim Self: Sorry Lynton, no prize for you. You made the same mistake as I did of not working with the whole problem statement. Beyond that, you did not submit an M expression for your interpretation, such as 0=1 or just plain 0. However, "untestible" is not the same as false which appears to be what you intended. The apparent contradictions in your preferred solution are very similar in nature to those in the problem statement.

Mark Sires: However, I would vote for Lynton to receive the prize for being the only one to notice the design specification would result in an unusable (or at least worthless) product. I could claim that I didn't notice it since I never saw the original post, but not only is it a weak excuse it would be untrue. Even if I saw the original post I would have focused on what I could do, rather than on what was asked for. This definitely gives me some insight I can use when developing specifications. In any case, to satisfy the requirement of a single expression would require one character, and here it is:

```
0
```

Alan Frank: I've been holding off on posting my solution because I haven't been able to prove that it works in all cases. However, I'm willing to bet that nobody will be able to construct a counterexample. I know that there are none within the portability limits (512 characters).

```
$L($TR(S,"QWERTYUIOP"))*$L($TR(S,"ASDGHJKL
"))*$L($TR(S,"ZXCVBNM"))=($L(S)**3/4)
```

The length is 80 characters.

Scott Jones: This may be the **shortest** solution, but not necessarily the most efficient (I'm pretty sure it will be slower than some other solutions already posted).

Maury Pepper: OK, it's time to lower the bar a little more. Jim Self's 131 character solution was the best so far. Here's one that's 123: (it's the same as my previous solution with the extraneous stuff removed.)

```
$TR(S,"ASDFGHJKLZXCVBNM","111111111.......")
?1(.NA,.NP,.AP)&($L($TR(S,"ZXCVBNM"))
*2=$L(S)!($L($TR(S,"ASDFGHJKL"))*2=$L(S)))
```

Jim Self: Very neat use of pattern match and $TRANSLATE, Maury. It looks like a solution and it checks out with my examples.

Lynton Blair: The following I did test as written (using an evaluation copy of Intersystems's OpenM, which I received yesterday). Actually, the color-coded GUI editor does make it much easier.

```
INITTEST ; initialize onlY once
  S R(1)="QWERTYUIOP",
  R(2)="ASDFGHJKL",R(3)="ZXCVBNM",a=1
  Q
TEST(S) ; return truth value for "Coding
Challenge" #1
  S f=0,Y=0
  I $G(a)="" Do INITTEST
  F N=1:1:3 Do
  . S S(N)=$TR(S,R(N))
  . I $L(S)'=$L(S(N)) S Y=Y+1,L(Y)=N
  . Q
  I Y=2 S f=$L(S(L(1)))=$L(S(L(2)))
  Q f
```

Ben Bishop:

```
Let Z be the number of chars from the
'third' row,
(2*X)*(X+Z)*(X+Z) ==> (2*X**3) +
(4*Z*X**2) + (2*X*Z**2)
is supposed to equal (2*X+Z)**3 / 4 or:

(2*X+Z)*(2*X+Z)*(2*X+Z)/4 ==> (2*X**3) +
(2*Z*X**2) + (X*Z**2) + (Z**3/4)

and the two sides should only be 'equal'
when: 2*Z*X**2 + X*Z**2 = Z**3 / 4

or when Z**3 - 4*X*Z**2 - 8*X**2*Z = 0 =
Z * (Z**2 - 4*X*Z - 8*X**2)
which is at Z=0 and the other two roots
can be solved by quadratic formula.
(with a=1, b=(-4x), c=(-8x^2)—now what was
that pesky formula?)
```

```
(-b +/- root(b^2 - 4ac) ) / 2a +4x +/-
root(16x**2 + 32x**2) / 2 = 0
```

One root is: `4x/2 + 4x/2*root(3) = 0` or `2x*(1+root(3)) = 0` which will never be true for non-zero real x

The other is:

```
 +4x/2 - 4x/2*root(3) = 0 or
 2x(1-root(3)) = 0   which will also
never be true for non-zero real x.
```

So, I don't believe there are any real solutions beyond Z=0, so your formula is correct...

Alan Frank: Unfortunately, most of these statements are not true. Starting from the end, it's not true that `$L(S)**3/4` is the maximum possible value of the product. For example, with the string "MUMPS," the left-hand side is 36, while the right-hand side, `$L(S)**3/4`, is 31.25. Rather, you need to provide that if they're equal, then the left-hand side must be of the form `$L(S)/2*($L(S)/2)*$L(S)`. I have tested this programmatically for values of `$L(S)` up to 512. I've been working on a general proof. It turns out that if there are any other solutions, then there are two numbers `U` and `V` such that if `W=U**2-(V**2)`, then `W`, `W-(U*V)`, and `W+(U*V)` are all perfect squares. In this case, `V#(8*3*7*19)=0` and `FOR Z=5,13,17,29,37 U#Z=0!(V#Z=0)!(U-V#Z)=0`. From this we can see `V>3000`, `U>16000`. I don't remember what this means about the size of any possible exceptional strings in the original problem, but it probably means they need to be over 3000 (at least) characters. I'll let you know more when I have it.

A Metathread Regarding the "Coding Challenge" Thread

There was also an online discussion regarding whether this topic should appear in the *M Computing* "Threads" feature:

Dan Baer: Valerie J. Harvey wrote: "By now those of you with *M Computing* subscriptions should have your October issue, including the *Threads* feature that a number of you helped with. I have already received a nomination for the next 'Thread'—the coding challenge topic. I also notice continuing attention to the Y2K topic. Are there any other preferences for a list topic to cover in *Threads*?"

I'd be disappointed if that kind of challenge (getting everything on one line) would be featured as a topic.

I don't see any need to continue that kind of coding, even for a challenge. I'd prefer that any examples used would be those that used structured coding.

James Self: It seems to me that that little challenge elicited the first bits of actual MUMPS code that I have seen for a long time on this newsgroup that had not been totally trivial or quite system specific. I think the topic is quite educational and this exercise elicited solutions which displayed a variety of different techniques and approaches to what appeared to be a very simple problem. It caused at least some of us to exercise skills that are not practiced every day. Among other things, I personally found that my math skills were much more degraded from lack of use than I thought. I would like to see more programming challenges and other topics that elicit concrete examples of MUMPS code. Perhaps Dan or someone else would like to submit a solution to Alan's challenge in the form of a function where the goal is to provide a well-structured source or a most efficient solution. We could then use this as a basis for discussion of "structured coding" or "good MUMPS code" or efficiency techniques and measurement.

Maury Pepper: Dan, I hope you're not worried that the "coding challenge" is in some way an example of how one SHOULD code. It's nothing more than an entertaining puzzle — kind of like those "white mates in two moves" chess puzzles. Maybe it's an outlet for those of us who just could never get enough brain teasers.

I would agree that "serious" space should not be

devoted to such matters, but it's not uncommon for a magazine to have a games section. And, sometimes it's such games that get students involved in using a new programming language. It helps one learn the intricacies of a language while having fun playing a game (i.e., solving a problem).

Dan Baer: Maybe it would be better to look at Y2K issues, or some other topic.

Jim Self: It might be interesting to read about specific measures that people on this list have taken to identify and correct Y2K problems. Has anybody found any really interesting or huge Y2K problems in their M systems? Do most people believe that they do not have a serious problem and what have you done to make sure? M

—Valerie J. Harvey Ph.D.

Marc Alan Asher	*etherware@aol.com*
Dan Baer	*mtrc@mcenter.com*
Ben Bishop	*aci@shore.net*
Lynton Blair	*lblair@ibm.net*
Alan D. Frank	*alf@world.std.com*
Antonio Gino	*gino@hawaii.edu*
Scott P. Jones	*scott@intersys.com*
Thomas Kiendl	*tck@dhzb.de*
Stefan Moernaut	*moernauts@spector.be*
Mark Sires	*msires@gte.net*
Maury Pepper	*mlp@cms-stl.com*
Mike Pohl	*mikedpohl@aol.com*
Jim Self	*jaself@ucdavis.edu*
Jim Vitek	*vitekj@MEDIAONE.NET*
