Puzzles are a source of learning and amusement for millions of people worldwide and have been for centuries. Crosswords, anagrams, math problems and geometrical puzzles serve as both recreation and tools to sharpen reasoning and vocabulary. In this spirit, this paper presents a series of questions which demonstrate the variety of the SAS software available and the power and complexity of the SAS system as a processing language. The puzzles below are a cross-section from centuries old tradition involving people and puzzles, enjoy and learn.

Puzzle 1: Titles and footnotes
You are producing a simple PROC PRINT listing and need to display the following title centered on each page:

LEVEL OF CO₂ PER SAMPLE

Can you do it using only the base SAS software?

Puzzle 2: Full-page report writing
You are trying to produce a double-column customized report using a DATA step. You want the report to utilize the entire page length (which has been defined through the statement OPTIONS PS=60;) so you’ve used the N=PS option on the FILE PRINT statement. Unfortunately, every time the first column of a page prints out there is an unexpected page break. This is the code being used:

OPTIONS PS = 60;
DATA NULL;";
SET DATASET;
FILE PRINT N_60 NOTITLES;
ROW = MOD (N_1, 60) + 1;
IF ROW = 1 THEN COLUMN = COLUMN - 1;";
IF COLUMN GE 60 AND ROW GE 60 THEN PUT PAGE;"
RUN;

This is an example of the print out:

-------- --------
x0x0x0x0x0x ----------
xAxAxAxAxAxAxAxAxAxAA x0x0x0x0x0x----------
xAxAxAxAxAxAxAxAxAxAA (continue for 50 more lines) x0x0x0x0x0x----------
xAxAxAxAxAxAxAxAxAxAA (page break) x0x0x0x0x0x----------
xAxAxAxAxAxAxAxAxAxAA (continue for 50 more lines) x0x0x0x0x0x----------
xAxAxAxAxAxAxAxAxAxAA (page break) x0x0x0x0x0x----------
xAxAxAxAxAxAxAxAxAxAA (etc.) x0x0x0x0x0x----------

What is the solution to your problem?

Puzzle 3: User-defined macro functions
You are coordinating a round-the-clock interactive data collection operation implemented through SAS software. There are six 4-hour shifts each day: midnight to 3:59 am, 4 am to 7:59 am, 8 am to 11:59 am, noon to 3:59 pm, 4 pm to 7:59 pm, and 8 pm to 11:59 pm. Each shift is divided into four 1-hour functional segments: set-up and preparation, primary data collection, secondary (follow-up) data collection, and wrap-up and reporting activities.

When an employee logs into the computer system, the software must automatically determine which function should be active at that time of day and execute the corresponding part of the interactive system.

Using SAS software macro capabilities, how would you implement a solution?

Puzzle 4: Macro variable resolution
Suppose you have defined a series of "linked" macro variables as follows:

%LET X0 = X1; %LET X1 = X2; %LET X2 = X3; %LET X3 = X4; %LET X4 = Xn; %LET Xn = Xn+1;

For the general case, how many &’s are required in an expression such as &X&X0 to obtain a final resolution of Xn?

Can you write a simple macro to generate the correct number of ampersands to obtain Xn?

Puzzle 5: Arrays
A survey form has two PARTS, A and B, each a rectangular matrix of variables. The variable names representing each cell are A1-A20 and B1-B12 as shown below. All cells (variables) marked with an asterisk should be null (missing) since these cells are shaded on the original questionnaire.

SURVEY FORM - RESPONDENT NAME: JOHN Q. PROGRAMMER

--- PART A -------------------------------
<table>
<thead>
<tr>
<th>Line</th>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A1</td>
<td>A2</td>
<td>A3</td>
<td>A4</td>
<td>A5</td>
</tr>
<tr>
<td>2</td>
<td>A6</td>
<td>*A7</td>
<td>*A8</td>
<td>*A9</td>
<td>A10</td>
</tr>
<tr>
<td>3</td>
<td>A11</td>
<td>A12</td>
<td>*A13</td>
<td>*A14</td>
<td>A15</td>
</tr>
<tr>
<td>4</td>
<td>A16</td>
<td>A17</td>
<td>A18</td>
<td>A19</td>
<td>A20</td>
</tr>
</tbody>
</table>

--- PART B -------------------------------
<table>
<thead>
<tr>
<th>Line</th>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B1</td>
<td>*B2</td>
<td>*B3</td>
<td>*B4</td>
</tr>
<tr>
<td>2</td>
<td>B5</td>
<td>B6</td>
<td>B7</td>
<td>B8</td>
</tr>
<tr>
<td>3</td>
<td>B9</td>
<td>B10</td>
<td>B11</td>
<td>B12</td>
</tr>
</tbody>
</table>

Below is a program which checks each of the key cells for a non-missing value and reports the part, line, column and value of each invalid entry. The data is stored in a dataset called SURVEY (one observations/respondent). The puzzle is to fill in the missing elements of the program - marked with ???(#)???
**Puzzle 6: Time manipulations**

You have the following list of activities occurring within a 24 hour period (on the same day):

<table>
<thead>
<tr>
<th>Activity</th>
<th>Start Time</th>
<th>End Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN</td>
<td>09:00:00</td>
<td>10:00:00</td>
</tr>
<tr>
<td>JUMP</td>
<td>09:00:00</td>
<td>11:00:00</td>
</tr>
<tr>
<td>SKIP</td>
<td>09:00:00</td>
<td>12:00:00</td>
</tr>
<tr>
<td>WALK</td>
<td>13:00:00</td>
<td>16:00:00</td>
</tr>
</tbody>
</table>

Write a program which will break the 24 hour period into hourly time blocks marking the duration of the activity.

Let **ACTIVITY** be a SAS dataset with variables **ACTIVITY**, **STARTIME** and **ENDTIME**. Create **HOUR1 - HOUR24** variables where 1 indicates the activity occurring in that hour.

As an added challenge, generalize the program using the SAS software macro capabilities. Allow parameters for an input dataset, an output dataset and the number of intervals in a 24 hour day (eg. 96 for fifteen minute intervals).

---

**Puzzle 7: Tricks of data manipulation (grouping observations without a grouping variable)**

Your client has requested the ability to print the observations in a PROC PRINT generated report into three-observation sets to permit easier reading across the page. That is, print three observations, then a blank line, then three more observations, another blank line, etc.

Using the data below, can you write a solution that eventually uses PROC PRINT?

<table>
<thead>
<tr>
<th>SAMPLE</th>
<th>LOCATION</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>EAST</td>
<td>100</td>
</tr>
<tr>
<td>B2</td>
<td>NORTH</td>
<td>57</td>
</tr>
<tr>
<td>A3</td>
<td>SOUTH</td>
<td>122</td>
</tr>
<tr>
<td>A4</td>
<td>SOUTH</td>
<td>93</td>
</tr>
<tr>
<td>A5</td>
<td>SOUTH</td>
<td>79</td>
</tr>
<tr>
<td>A6</td>
<td>EAST</td>
<td>112</td>
</tr>
<tr>
<td>A7</td>
<td>NORTH</td>
<td>109</td>
</tr>
<tr>
<td>A8</td>
<td>EAST</td>
<td>94</td>
</tr>
<tr>
<td>A9</td>
<td>NORTH</td>
<td>78</td>
</tr>
</tbody>
</table>

Can you make it general enough to change the number of observations per grouping?

Can you make it general enough to handle ANY number of variables of ANY type?

---

**Puzzle 8: Tricks of procedure output (sorting without PROC SORT)**

You have a data set named **SALES** with sales information for three local showrooms. It has three variables: **SHOWROOM**, **MODEL** and **AMOUNT** (total number of sales of the model).

Data set **SALES**:

<table>
<thead>
<tr>
<th>SHOWROOM</th>
<th>MODEL</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Av.</td>
<td>Basic</td>
<td>12459</td>
</tr>
<tr>
<td>Central Av.</td>
<td>Advanced</td>
<td>9367</td>
</tr>
<tr>
<td>Central Av.</td>
<td>Intermediate</td>
<td>3190</td>
</tr>
<tr>
<td>Washington St.</td>
<td>Basic</td>
<td>16332</td>
</tr>
<tr>
<td>Washington St.</td>
<td>Advanced</td>
<td>5010</td>
</tr>
<tr>
<td>Washington St.</td>
<td>Intermediate</td>
<td>2916</td>
</tr>
<tr>
<td>Beach Blvd.</td>
<td>Basic</td>
<td>2334</td>
</tr>
<tr>
<td>Beach Blvd.</td>
<td>Intermediate</td>
<td>945</td>
</tr>
</tbody>
</table>

You need to produce a report which lists the number of sales per model in the following manner:

1. List the following variables in descending order:
   a. first list **MODELs** by total number of sales,
   b. then list **SHOWROOMs**, within each **MODEL**, by the total number of sales across all models.
2. The overall number of sales per model is listed at the end of each model's section.
3. Since the number of sales are typically more than 1000 per model per showroom, make sure the sales counts are printed with commas for easier reading.

There is a solution using just two SAS procedure steps, neither of which is a PROC SORT. What are these procedures?

---

**Puzzle 9: Tricks of DATA manipulation (multiple data sets)**

You have been given the following two SAS data sets, where **K1** and **K2** uniquely determine an observation in each data set. Assume both sets are sorted by **K1** and **K2**.

<table>
<thead>
<tr>
<th>Data set A</th>
<th>Data set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>K2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Write a SAS program to form the data set **C** where:

1) **K1** values appear in **C** only if the values exist in both **A** and **B**, and
2) **K2** is a corresponding value which may be from **A**, **B** or both.

Can you write a SAS program to form the data set **C** where:

<table>
<thead>
<tr>
<th>Data set C</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

45
Solution to puzzle 1: Titles and footnotes

While subscripts (and superscripts) are directly possible in the SASGRAPH® software using the M= option with a title, the best way is to utilize the %SCAN function containing the same number of characters guarantees alignment of the centered output lines.

```
TITILE1 "LEVEL OF CO PER SAMPLE";
TITILE2 " 2 ";
```

Solution to puzzle 2: Full-page report writing

There should be a trailing @ at the end of the PUT statement. As it stands now, a new page is generated after writing row 60 because the "row pointer" is being advanced from 60 to 61, which is past the end of the page. Think of the " at the end of the PUT statement as a carriage return; by placing the trailing @ before it, the semicolon is treated as just a carriage return, keeping the "row pointer" on row 60.

Solution to puzzle 3: User-defined macro functions

One way to approach the problem is to utilize the automatic SAS macro variable &SYSTIME which contains the time (in military format. i.e., 0:00 through 23:59) at which the SAS system was invoked. By identifying the hour portion of &SYSTIME using the %SCAN function, we can execute the proper module.

```
%MACRO STARTUP;
LET HR=%SCAN(&SYSTIME,1,:);
%IF &HR=0 OR &HR=4 OR &HR=8
%THEN 'LET HR='00;
%ELSE 'LET HR='SCAN(&SYSTIME,1,:);
%END;
%INST;
%MEND STARTUP;

%MACRO AMPERS(N);
%LET AMPERS="\"&AMPERS,\"X\";
%LET AMPERS="\"&AMPERS,\"X\";
%LET AMPERS="\"&AMPERS,\"X\";
%MEND AMPERS;

%MACRO MOD(NUMBER, MODULUS);
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%MEND MOD;

%MACRO MOD(NUMBER, MODULUS);
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%MEND MOD;

%MACRO STARTUP;
%LET HR=%SCAN(&SYSTIME,1,:);
%LET SEG=MOD(&HR,4);
%IF &SEG=0 THEN VDO; %SETUP; %END;
%ELSE 'IF &SEG=1 THEN VDO; %COLLECT1; %END;
%ELSE 'IF &SEG=2 THEN VDO; %COLLECT2; %END;
%ELSE 'IF &SEG=3 THEN VDO; %REPORTS; %END;
%MEND STARTUP;
```

Solution to puzzle 4: Macro variable resolution

The general solution requires $(2^n-1)$ ampersands to resolve to $X_n$. This is due to the collapsing of double ampersands to single ampersands performed by the macro processor when resolving indirect macro variable references. For example, a representation of the resolution to $X_3$ is illustrated below (in reverse order).

```
X_3 <--- &X_2 <--- &X_1 <--- &X_0
```

To generate $(2^n-1)$ ampersands, it is relative efficient to initialize a macro variable as a single & and iteratively append the current value to itself. However, we must take care that multiple ampersands do not resolve too early in the process; thus, we will utilize the %QUOTE function.

After the final $(n^{th})$ iteration, we will have $(2^n-1)$ ampersands. In order to remove one of the ampersands, we choose the %SUBSTR function, but as indicated earlier, we must not use it too early. In fact, we don't invoke %SUBSTR until the exact point at which we wish to obtain the ultimate resolved value.

```
%MACRO AMPERS(N);
%GLOBAL AMPERS;
%LET AMPERS=4;
%DO = 2 ND &N;
%LET AMPERS=QUOTE(AMPERS) QUOTE(AMPERS);
%END;
%MEND AMPERS;

%MACRO MOD(NUMBER, MODULUS);
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%MEND MOD;

%MACRO MOD(NUMBER, MODULUS);
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%LET MODULUS=%MODULUS;
%MEND MOD;
```

Solution to puzzle 5: Arrays

Substitutions for ???'s

1. VNAME
   Function puts the variable name from current array position into the variable NEWVAR.

2. INT;
   (ELEMENT-1) / INPUT(PART, $PTCOL1.2.1 )
   ) +1

3. MOD;
   (ELEMENT-1) / INPUT(PART, $PTCOL1.2.1 )
   ) +1

46
Solution to puzzle 6: Time manipulations

Basic solution:

```
DATA TIMES (DROP=I INTVL);
RETAIN INTVL 3600;
J,o
SECONDS IN 1 HOUR
,f,o
SET ACTIVITY;
ARRAY _TS (,o) TI-T24;
DO 1=1 TO DIM(_TS);
__TS (I) (STARTIME < I * INTVL ) AND
(ENDTIME > (I-1) * INTVL );
END;
RUN;
```

Generalized solution:

```
%MACRO MARKTIME(INS" _LAST_, OUTDS--_DATA_, NUMINTVL=24 I:
DATA oI;OOTDS (DROP-oI INTVLI:
RUN;
RETAIN INTVL EVAL(8li400 I
'NUMINTVL);
SET UNDS;
ARRAY _TS (*l T1 - T'NUMIN'lVL;
DO I =' 1 TO lNUMIN'lVL;
_TS(I) (STARTIME <I IN'lVL) AND
(ENDTIME > (I-1) * INTVL);
END:
%MEND MARKTIME:

THIS MACRO CALL GENERATES IDENTICAL CODE
TO
THE SOLUTION ABOVE

%MARKTIME (lNDS"ACTIVITY, OUTDS TIMES , NUMINTVL = 241
PROC PRINT
DATA TIMES
TITLE 'ASSIGNED TIME INTERVALS';
VAR ACTIVITY STARTIME ENDTIME T1-T24;
RUN;
```

Solution to puzzle 7: Tricks of Data manipulations (grouping observations without a grouping variable)

One way to solve this problem is to literally insert "blank" records after every n observations. This blank record will have a missing value for every variable.

In the following code, consider these points:

1. an OPTIONS MISSING=' ' statement is used so that missing numeric values do not print;
2. to handle any types of variables, use the special_NUMERIC_ and_CHARACTER_keywords in an ARRAY;
3. to cover the case where all variables are numeric or character, define a "dummy" variable of each type, then DROP them from the output data set;
4. to detect every nth observation, the MOD function is useful to identify when to insert the blank record.
5. a MACRO variable (&GRPSIZE) is used to specify the number of records per group.

```
OPTIONS MISSING=" ";
%LET GRPSIZE=3; /* GROUPING DESIRED */
DATA AFTER (DROP= DUMMYN __ DUMHMGC_);
SET BEFORE;
RETAIN DUMMYN 0 DUMHMGC_ ;
OUTPUT ; /* ORIGINAL DATA OBSERVATION */
IF MOD(N, &GRPSIZE)=0 THEN
DO ; /* CHECK GROUPING SIZE */
ARRAY NUNS (*) NUMERIC_;
ARRAY CHARS (*) CHARACTER ;
DO I=1 TO DIM(NUNS); NUNS(I)=. END;
DO I=1 TO DIM(CHARS); CHARS(I)=. END;
OUTPUT ; /* OUTPUT BLANK RECORD */
END ; /* CHECK GROUPING SIZE */
RUN;
PROC PRINT DATA=AFTER NOOBs;
RUN;
```

The PROC PRINT will generate the following output:

```
<table>
<thead>
<tr>
<th>SAMPLE</th>
<th>LOCATION</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>EAST</td>
<td>100</td>
</tr>
<tr>
<td>A2</td>
<td>NORTHEAST</td>
<td>57</td>
</tr>
<tr>
<td>A3</td>
<td>SOUTH</td>
<td>122</td>
</tr>
<tr>
<td>A4</td>
<td>SOUTH</td>
<td>97</td>
</tr>
<tr>
<td>A5</td>
<td>SOUTHWEST</td>
<td>79</td>
</tr>
<tr>
<td>A6</td>
<td>EAST</td>
<td>112</td>
</tr>
<tr>
<td>A7</td>
<td>NORTHEAST</td>
<td>109</td>
</tr>
<tr>
<td>A8</td>
<td>EAST</td>
<td>91</td>
</tr>
<tr>
<td>A9</td>
<td>NORTHEAST</td>
<td>78</td>
</tr>
</tbody>
</table>
```

Solution to puzzle 8: Tricks of procedure output (sorting without PROC SORT)

The two procedures are PROC FREQ and PROC PRINT with the following options and parameters:

**PROC FREQ:** Using the ORDER=FREQ option on a two-way table (with OUT=) will output a data set with the cells in both directions sorted by the total count in each direction separately (item #1). That is, since the overall count for MODEL=BASIC is the largest, BASIC is the first value of MODEL in the output data set and, within each MODEL, SHOWROOM=CENTRAL AVE. appears first because that SHOWROOM value has the highest overall count. Also, don't forget to use the WEIGHT statement for the AMOUNT to indicate counts.

```
PROC FREQ DATA=SALES ORDER=FREQ;
TABLE MODEL*SHOWROOM/NOPRINT OUT=FREQDATA;
WEIGHT AMOUNT;
RUN;
```

**PROC PRINT:** Using the NOTSORTED option on the BY statement, the “groups” for MODEL can be printed exactly as output by PROC FREQ. The SUM=SUMBY statements provide the totals for each model (item #2). The numbers are easily displayed with commas by using the COMMA. format (item #3).

```
PROC PRINT DATA=FREQDATA;
FORMAT COUNT COMMA .;
BY MODEL NOTSORTED;
SUM COUNT;
ID MODEL; VAR SHOWROOM;
RUN;
```

The PROC PRINT will generate the following output:

```
<table>
<thead>
<tr>
<th>MODEL</th>
<th>SHOWROOM</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC</td>
<td>CENTRAL AVE.</td>
<td>12,159</td>
</tr>
<tr>
<td></td>
<td>WASHINGTON ST</td>
<td>14,332</td>
</tr>
<tr>
<td></td>
<td>BEACH BLVD.</td>
<td>2,334</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>BASIC</td>
<td>31,259</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>ADVANCED</td>
<td>CENTRAL AVE.</td>
<td>9,367</td>
</tr>
<tr>
<td></td>
<td>WASHINGTON ST</td>
<td>6,010</td>
</tr>
<tr>
<td></td>
<td>BEACH BLVD.</td>
<td>2,331</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>ADVANCED</td>
<td>16,488</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>CENTRAL AVE.</td>
<td>3,190</td>
</tr>
<tr>
<td></td>
<td>WASHINGTON ST</td>
<td>2,916</td>
</tr>
<tr>
<td></td>
<td>BEACH BLVD.</td>
<td>945</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>7,051</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>54,664</td>
<td></td>
</tr>
</tbody>
</table>
```
Solution to puzzle 9: Tricks of DATA manipulation (multiple data sets)

The first step is to create not one but two data sets: one containing the merged data set A and data set B, another containing the common K1 values between data set A and data set B. The MERGE statement with the IN= parameter, the SUM statement and the LAST=.BYVARIABLE are used to create these two data sets.

```sas
DATA AB (DROP = _ACNT _BCNT) WANTED(KEEP = K1);
    MERGE A (IN= __INA) B(IN= __INB);
    BY K1 K2;
    IF __INA THEN _ACNT + 1;
    IF __INB THEN _BCNT + 1;
    OUTPUT AB;
    IF LAST.K1 THEN
      DO;
      END;
    IF _ACNT AND _BCNT THEN OUTPUT WANTED;
      _ACNT = 0;
      _BCNT = 0;
RUN;
```

The MERGE statement uses the IN= parameter to identify the original data set for the observations in the merged data set AB. With these IN= flags, two separate SUM statements (or counters) were set up for the two data sets (_ACNT + 1; _BCNT + 1). Which, in turn, are used in a DO loop to determine what observations are output to the WANTED data set. Do you see what is going on with this IF statement conditional expression? Why do you think _ACNT and _BCNT are reset to 0 after an observation is output to the WANTED data set? Do you think the KEEP option on the WANTED data set name is important? (Take a look at the next step.)

```sas
DATA C;
    MERGE AB WANTED (IN= _WANTED);
    BY K1;
    IF _WANTED;
RUN;
```

We create our final data set C by merging our combined AB data set with the WANTED dataset and a subsitting IF. Do you think the merge order is important in this step?

Acknowledgments: The authors wish to thank the staff of the ARC Professional Services Group for contributing to the puzzles presented in this paper. Also, special thanks to Ian Whitlock for beginning the seed for this paper with his "Puzzles Corner".

The authors may be contacted at:

ARC Professional Services Group
601 Indiana Ave NW, Suite 510
Washington, D.C. 20004
(202) 737-2666

SAS and SAS/GRAPH are registered trademarks of SAS Institute Inc., Cary, NC, USA.