DEMYSTIFYING PROC FORMAT
HALLETT GERMAN, GTE LABORATORIES

Introduction

After having grasped its fundamentals, most SAS® users will spend a considerable amount of time learning ways to control and enhance report output. Formats -- both system-supplied and user-defined provide an easy and flexible means of determining how values will be displayed or printed.

This paper has two aims: a) to provide beginning SAS® users with an easy-to-understand approach for learning formats and b) to illustrate how to use this approach for enhancing your SAS reports.

What Are Formats Anyway?

Formats are a set of instructions to tell the SAS system how to read values (generally called INFORMATS) and how to display or print values (generally called FORMATS not OUTFORMATS). Another way of thinking about formats is as a table lookup -- finding a value in another file based upon the value in the main file. Let's look at a brief example. Suppose you wanted to order a pizza but only remembered the name of the pizzeria and not the phone number. For most of us this would not be a problem -- we would "look up" in our phone book "phone number table" the name of the pizzeria and next to it would be the "associated value" of telephone number. Thus, formats are a means of transforming input or output values to have the desired numeric or character values. This can be done by changing the data type, by using an associated value, or recoding a continuous variable into a discrete (distinct categories) one.

Formats -- A Report Writing Tool

1. Although a FORMAT Statement can be used with all procedures, some common uses are:
   * CALENDAR, FORMS (limiting length of output values)
   * GCHART, GPLOT, CHART, PLOT (creating axes or output range/values)
   * MEANS, PRINT (determining output range/values on all report variables)
   * SUMMARY (limit number of class variables)
   * FREQ, TABULATE (change display of output)

2. The following procedures will give information about formats and format libraries.
   * CONTENTS (can display informat/format information about SAS datasets)
   * FMTLIB (list/output format library directories
     this is in supplemental library)
   * PDS (list/delete all or some members of a format library)
   * PDSCOPY (copy formats between format libraries and to list the relative size of a format)

With this brief listing, you can sense the power and flexibility of formats. Whether it is changing the data values for graphs, reports, or statistical analysis, formats can easily produce the desired result.

Format Statement vs. Format Procedure -- What Gives?

Up to this point, I have used the term format quite loosely to include both the FORMAT statement and the FORMAT procedure. But they both do different things:

* Format Statement - With PUT and ATTRIB statements, is one of three ways to temporarily or permanently associate a format/informat with a variable. If the FORMAT statement appears in the DATA step, then the association is permanent until reassigned. If the FORMAT statement appears in the PROC step, then the association lasts only for the run of the PROC.
Notes:

1. A FORMAT statement can contain more than one format — such as many variables associated with one format or different variables associated with different formats.

2. The last FORMAT or ATTRIB statement that a variable appears in will determine the current variable-format association.

3. An important point is PERMANENTLY ASSOCIATED DOES NOT MEAN PERMANENTLY STORED. For permanent SAS datasets the user-defined format must always be accessible (whether or not it is used).

4. If a FORMAT statement is given so that a variable has no format associated with it, then a default format (BEST. or $w.) will be associated with the variable. An example is FORMAT X; - if X is numeric then it will be displayed in BEST format, otherwise $w. format.

* Format Procedure — Used to create temporary (the default) or permanent formats that are compiled into executable members of a format library. NOTE: The members do not contain the PROC FORMAT statements that created it).

System-Supplied Formats

SAS system includes nearly 40 system supplied informats/formats to help you read and write variables. These formats/formats can be divided into four categories: 1) numeric, 2) character, 3) date/time, and 4) column-binary (informats only). With these formats you can 1) make a variable more user-presentable (e.g. DATE7. will take an internal date and transform it to the easier-to-understand 20MAY87) or 2) let you read or write a variable in a different manner (such as trailing zeros are blanks - B23. will read '55 ' as '550').

All formats (whether system-supplied or user-defined) have a certain syntax and components. Figures 1 and 2, illustrate the components of formats (figures are at end of the article).

The following are the major points concerning system-supplied formats:

1. Applies to all formats (whether system-supplied or user-defined):
   a. Format names follow the general rule for SAS names - a) 1-8 characters (including a dollar sign for character formats), b) format name never ends with a number, and c) cannot be a reserved SAS word. Character formats always start with a $ (dollar sign).
   b. Formats, when specified where format names are expected, always include one period. For all non-decimal formats, it is at the end of the format (decimal formats never require two periods). WITHOUT THE PERIOD, SAS software WILL NOT RECOGNIZE IT AS A FORMAT! The exception to this are the VALUE, INVAL'JE, and PICTURE statements of PROC FORMAT.
   c. decimal formats (example: 10.2). The ten includes a maximum of seven non-decimal digits, one decimal point, and two decimal digits). If the input value includes a decimal point, then the original position is maintained. Else, if a decimal format is used, the SAS system divides the number by the number of decimal digits (in this case - 2).
   d. decimal points for numeric formats are optional (and not applicable to character and many date/time formats.)
   e. FMTERR/NOFMTERR, the system option, determines whether or not SAS software gives an error message if a format is not found associated with a variable in a format statement. FMTERR (the default for all operating systems) will give the error message while NOFMTERR will use a default format instead ($w. or $w.) and give no warning.

2. The following applies to system-supplied formats only:
   a. Each informat/format has its own default minimum length and justification.
   b. You can create your own "homemgrow" system-supplied formats and informats. Check SAS Technical Reports P-162 and P-169 (VMS) "User-Written Functions, CALL routines, Informats, and Formats for the Version 5 SAS System".
   c. There are more formats than informats available.

This can only be a brief introduction on system-supplied formats. For further information, the reader is encouraged to look at the "SAS Informats and Formats" Chapter of the SAS Basics Book.
Invoking SAS Formats

Figure 3 reviews the various means of invoking SAS formats.

The following summarizes the major points about each of these statements:

* ATTRIB • can be used to associate a format, informal, length or label with a variable. The variable-format association can be overridden by an ATTRIB, INFORMAT, or another FORMAT statement. One ATTRIB statement can have many associations and follows the rules listed above for decimal formats. ATTRIB be used in either the DATA (permanent association) or PROC (association run of the PROC.) steps.

* FORMAT - Similar to ATTRIB but for formats only. Generally used in both DATA and PROC steps.

* INFORMAT - Similar to ATTRIB but for formats only. Generally used in DATA step and a few procedures that input data.

* INPUT - Can specify the informal directly in the INPUT statement. Formatted input statements are generally easier to follow, and give you flexibility in reading data (in terms of variety, the ability to read grouped format lists, and use format modifiers e.g. and ). INPUT is in the DATA step only.

* PUT - Similar to INPUT but used in the DATA step only to write lines to the SAS log or any file. Do not confuse the INPUT and PUT statements with the PUT and PUT functions which transform a value to the desired informal or format.

User-Defined Formats

The real power of formats is the ability to generate your own using PROC FORMAT. User-defined formats are placed into two categories:

value-labeled formats - Places an easy to understand text label (so instead of , FEMALE will be printed) for both character and numeric formats.

picture formats - Specifies a template containing the desired combination of character and numerics (for numeric data only).

Each of these will be discussed.

Value-labeled Formats

Figure 4 gives examples of value-labeled formats.

Note the following about value-labeled formats:

1. The format creation has three parts: a) the PROC FORMAT statement, b) the value statement with the name of the format, and c) the format values (where input value = output value). The three parts can be on one or more lines. If all input values are alphanumeric or special characters, then it is a character format. Otherwise, it is a numeric format. Output values should almost always be in single quotes (unless the SAS option TEXT is in effect).

2. More than one value-labeled or picture formats can appear in the same PROC FORMAT statement. But only one format per VALUE or PICTURE statement.

3. Value names follow the general rules for SAS names (listed previously). Values can be the same name as the associated variable.

4. The input value (left of equal sign) - (such as 1 or 'A') can be a single value, a range of values (denoted by dashes) or a series of ranges/non-sequential values.

5. To avoid problems, do not use overlapping ranges or values:

    VALUE RETIRE 0-65 = 'WORKING'
    65-100 = 'RETIRED';

If a value of 65 is encountered, then the SAS system will associate the value with the first range found.

6. Input values can use keywords LOW, HIGH, or OTHER in specifying ranges.

7. Output values can be up to 40 characters and should always be in single quotes.

8. If the variable value is not one of the listed input values, then the default format and justification is used.
Picture Formats

Picture formats allow you to transform input value to certain combinations of character and numerics. This means that you can a) eliminate leading zeros, b) format comma and decimal punctuation, c) specify fill characters, d) represent negative numbers, and e) allow values to be multiplied by a constant before being formatted.

9. Some procedures will only print the first eight or sixteen characters of the format.

To fully understand how picture formats work, I encourage you to read pages 922-923 of the SAS BASICS book on "Picture Logic." However, here is a quick review of picture syntax.

1. The format creation has three parts; a) the PROC FORMAT statement, b) the PICTURE statement with the name of the format, and c) the format values (where input value = output value). The three parts can be on one or more lines. Picture statements can only be numeric formats.

2. Otherwise follows the same rules as value-label formats in syntax and names. Keywords LOW, HIGH, OTHER are also available.

3. Output formats follow the following rules:
   a. Should almost always be in single quotes unless the global option TEXTB2 is in effect.
   b. Is really a template of which combination of numbers and letters to print. If you use digits 1-9 in your template, then leading zeros will be retained. Otherwise a 0 digit in your template will not retain leading zeros.
   c. In addition to options in the PROC FORMAT and PICTURE statements, output formats have their own options which are designated by parentheses. This is NOT true for value-labeled formats.
   d. Output formats cannot contain a message (alpha) character -- Use the PREFIX or FILL options (where they are always in single quotes). PREFIX and FILL characters must fit within the picture and ARE NOT ADDITIONAL CHARACTERS. If the format isn't large enough, the prefix is dropped.
   e. Output formats can be no longer than 24 characters.

Figure 5 shows some the various uses of picture formats:

In example 1, Text messages (using the NOEDIT picture option) were generated and did not retain leading zeros (except for one significant digit and the two decimal digits). Example 1 also used fill characters and a prefix (for negative numbers). Examples 2 and 3 show how to add a suffix right in the picture (output format). Example 3 also uses a multiplier option which multiplies before formatting data. This option is used to replace decimal points with an easy-to-understand suffix (such as K or M). The most common use of pictures is generating phone numbers. Steve Schultz, in last year's paper, presented a useful way of doing this (Readers should also look at Steve's phone number picture with NOEDIT). Example 4 reproduces Schultz's format. Finally, example 5 is there to make a point - you can make a picture format into a value labeled one as much as you want (using NOEDIT).

Reading Unknown Values

One of the four options available for value-labeled or picture formats (MIN, MAX, DEFAULT, and FUZZ), is FUZZ --the fuzz factor. The fuzz option gives a range of values to match the format value — it does not format (round the result) but does allow you to read unknown values. Unfortunately, FUZZ is not available for formatting of pictures (which is too bad because picture truncates not rounds numbers. See the Langston reference for further information. The following is a brief example of the FUZZ option:

```
PROC FORMAT;
   VALUE ZZZ (FUZZ=.2) 1=",0," 2="6" 1="C" OTHER = "OUT OF RANGE;
```

This means if a number is within .2 of an input value, it is seen as a match. Thus 2.1 is a B and 2.9 is a C.

Format Libraries and Utilities

As mentioned before, formats are stored as executable members of format libraries. These formats can be saved in a permanent library to be reused as often as needed (without recreating the PROC FORMAT code every time). With
permanent format libraries, you or your department can standardize formats (such as having one format for age or "yes/no" questions). The means of generating these format libraries is dependent on the operating system so check the Appendix "Temporary or Permanent Formats" in the PROC FORMAT chapter of the SAS Basics book or the appropriate SAS Companion for your Operating System.

The beginning of this paper discussed four SAS utility procedures to perform format library operations. FMTLIB can also be useful in recreating formats (or porting to a different operating system). Macros can be used to have self-generating formats. The Ray and Levine or Merlin references provide examples on how to do this.

**Advantages of Formats/Uses/Etc..**

1. If less than 10,000 input values are being used, then format is extremely fast for table lookups (and frees your SAS program from many IF statements).
2. Flexible - can do lookups on different formats in 1 dataset or have conditional lookups.
3. Format libraries provide a chance to standardize/minimize code.
4. You can create your own formats/informats or use the system-supplied ones.
5. You can use formats to "clean up" almost all SAS procedure output (i.e. a smaller length to get more in a report).
6. The dataset does not have to be sorted to perform lookup (find match).
7. You don't have to add another variable to a dataset - use a format.
8. Provides efficient storage and use of information (as compared to IFS.)
9. An easy means to do numeric-character or character-numeric conversions.
10. Can make the range of the output value as large or as little as you want.

**Disadvantages/Limitations/Non-Uses/Etc.**

1. No FUZZ option for Picture.
2. Only 40 characters (and 8-16 characters for some procedures) are allowed. Character output values has a maximum of 16 characters and pictures a maximum of 24 characters.
3. The cost of CREATING the format increases linearly to the size of the lookup table.
4. No SAS book just on Format usage (similar to TABULATE or macros).
5. The SAS system limits the number of value labels to 32,760 (approximately).
6. Column headings longer than eight characters in PROC FREQ are truncated (even with special formats).

**Proc Format – A Report Writing Tool Revisited**

The remainder of this paper will discuss some of the common uses for enhancing your reports with PROC FORMAT and the FORMAT statement:

1. Improving the appearance of your reports by making your data more presentable.

This should not be a surprise to you since it has been mentioned throughout the paper. Both value-labeled and picture formats can be used for this purpose. Looking at a PROC PRINT before and after being formatted should make this point clear:

**BEFORE FORMAT**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SEX</th>
<th>NET EARNINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAL</td>
<td>M</td>
<td>100000</td>
</tr>
<tr>
<td>SOPHIA</td>
<td>F</td>
<td>1000000</td>
</tr>
</tbody>
</table>

**AFTER FORMAT**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SEX</th>
<th>NET EARNINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAL</td>
<td>MALE</td>
<td>$0.1M</td>
</tr>
<tr>
<td>SOPHIA</td>
<td>FEMALE</td>
<td>$1.0M</td>
</tr>
</tbody>
</table>

The code to do this:

```sas
PROC FORMAT;
VALUE $SEX
'M' = 'MALE'
'F' = 'FEMALE'
PICTURE EARN
LOW-HIGH = $'09.9M'
(PREFIX = '$' MULT=.0001);
```

1266
data step statements go here
PROC PRINT;
VAR NAME SEX NETEARN;
FORMAT SEX $SEX. NETEARN EARN.;

Formats can enhance results from almost all SAS procedures, but is particularly useful for CALENDAR, FREQ, PRINT, and TABULATE.

2. Reduce the field width of a field when printed in a report. This can be very useful in procedures (such as PRINT) where space is determined by the maximum length of a value. Other possible procedures are FREQ, TABULATE, and FORMS. A simple example - using a two character state abbreviation instead of the entire name.

3. Recoding a variable from a continuous listing to one with discrete categories or creating a customized range from an existing one. This is useful for PROC FREQ (especially Crosstabs) and TABULATE. An example with a single frequency will illustrate:

SAMPLE NUMBER OF ACCIDENTS PER YEAR IN MASSACHUSETTS

```
ACCUM.  NUM  FREQ   PERCENT  CUM  CUM
       0    15    15.0   15.0   15.0
  1-2    10    10.0    70.0   16.0
>2     50    50.0   100.0   100.0
```

The outcome is only 3 categories instead of 5-10. For crosstabulations, this means using far less memory and easier to understand results.

4. Data Validation is a common technique that determines what observations a report may contain. To validate that a file has all observations from the Northeast:

```
PROC FORMAT;
VALUE REG 1 = 'NORTHEAST' ;
DATA TESTIT;
INPUT REGION SALES;
IF PUT(REGION REG.) NE '1' THEN
   bad record message/write record location
ELSE
   good record message
```

Pages 209-210 of the SAS Applications Guide has a detailed example of formats as a data validation tool. A brief aside - SAS Release 5.17 will support both character and numeric invalues for all operating systems. A Birds of a Feather Session at SUGI 13 will go into greater detail about this area.

5. Change the appearance of row and column headings. This is most useful for PROC FREQ (and a lesser extent for TABULATE). In crosstabulations (two dimensional frequencies), the column heading is divided into two lines after the first eight characters. To relieve this problem (characters longer than 16 characters are truncated), special formats for column headings are suggested. See Figure 6 for an example of this use.

6. Use PROC FORMAT to match and merge two unsorted files (a large one to a smaller one).

7. Limit the number of classification values for a Proc Summary.

8. Finally, PROC FORMAT can enhance your graphics output:
   a. GMAP - specify regions or range of an analysis variable (change a continuous variable into discrete categories)
   b. GPLOT - determining by variables for multiple plots, determining names of multiple plots (for legend), determining axes (especially date/time).
   c. GCHART - format block text in block charts and pie pieces text in pie charts.

References:
Henderson, Don, "Table Lookup Techniques", Proceedings of the Seventh Annual SUGI Conference 1982, pp. 792-6
Ray, Craig and Howard Levine "Efficient Use of Table Lookup Procedures" Proceedings of the Twelfth Annual SUGI Conference 1987, pp. 41-2


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

Figure 1: Numeric, Column-Binary and Some Date-Time Formats

<table>
<thead>
<tr>
<th>Description</th>
<th>Root</th>
<th>Width</th>
<th>Decimal</th>
<th>Entire Format</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print Leading Zeros</td>
<td>Z</td>
<td>8</td>
<td>1</td>
<td>28.1</td>
<td>1234.1</td>
<td>0012341</td>
</tr>
</tbody>
</table>

Figure 2: Character, Some Numeric, and Date-Time Formats

<table>
<thead>
<tr>
<th>Description</th>
<th>Root</th>
<th>Width</th>
<th>Decimal</th>
<th>Entire Format</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Character</td>
<td>$</td>
<td>3</td>
<td>N/A</td>
<td>3.0</td>
<td>ASC</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Figure 3: Means of Calling SAS Formats

<table>
<thead>
<tr>
<th>Process</th>
<th>Reading Values</th>
<th>Writing Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statements</td>
<td>INPUT NAME;</td>
<td>PUT NAME;</td>
</tr>
<tr>
<td>Comments</td>
<td>Input with default format</td>
<td>Put with default format</td>
</tr>
<tr>
<td></td>
<td>INPUT NAME 20;</td>
<td>PUT NAME 20;</td>
</tr>
<tr>
<td></td>
<td>INFORMAT NAME 20;</td>
<td>FORMATT NAME 20;</td>
</tr>
<tr>
<td></td>
<td>INPUT NAME;</td>
<td>PUT NAME;</td>
</tr>
<tr>
<td></td>
<td>ATTRIB NAME LENGTH=20;</td>
<td>ATTRIB NAME LENGTH=20;</td>
</tr>
<tr>
<td></td>
<td>INFORMAT = $;</td>
<td>FORMAT=$;</td>
</tr>
<tr>
<td></td>
<td>INPUT NAME;</td>
<td>PUT NAME;</td>
</tr>
</tbody>
</table>

1268
Figure 4: Value-labeled format examples

1. PROC FORMAT;
   VALUE $TOPPINGS
   'A' = 'ANCHOW'
   'B' = 'MUSHROOM'
   'C' = 'ONION'
   'D' = 'PEPPERS'
   OTHER = 'COMBINATION';
   FORMAT PIZZA $TOPPINGS.

2. PROC FORMAT;
   VALUE AGERNG
   0-12 = 'KID'
   13-21 = 'TEEN'
   22-HIGH = 'ADULT';
   FORMAT AGE AGERNG.

3. PROC FORMAT;
   VALUE PARTY
   '0' = 'PEPPERS'
   OTHER = 'COMBINATION';
   FORMAT P PARTY.

4. PROC FORMAT;
   VALUE $GRAOE
   'A' = 'EXCELLENT'
   'B' = 'GOOD'
   'C' = 'FAIR'
   'U','F' = 'SUMMER SCHOOL';
   FORMAT GRADE $GRADE.

Figure 5: Uses Of Picture Format

<table>
<thead>
<tr>
<th>Use</th>
<th>Code</th>
<th>Output</th>
</tr>
</thead>
</table>
| 1. Changing Leading Characters   | PROC FORMAT;
                                           PICTURE DOLL
                                           LOW-1E5 = 'OVERFLOW (NOEDIT)
                                           -99999.99-0 = '000,000.00'
                                           (PREFIX='-', FILL='')
                                           0-99999.99 = '000,000.00' (FILL=''
                                           1E6-HIGH = 'OVERFLOW (NOEDIT); |
|                                  |                                                                        | 10,000                        |
| 2. Adding Plus or Minus Signs    | PROC FORMAT;
                                           PICTURE SIGNX
                                           LOW-0 = '000,000.00'
                                           0-HIGH = '000,000.00' (FILL=';') |
| (+,- in picture)                 |                                                                        | 8,444.90+                     |
| 3. Add suffix and/or multiplier  | PROC FORMAT;
                                           PICTURE TH
                                           0-HIGH = '000,000' (MULT=.001); |
|                                  |                                                                        | 8,444.90+                     |
|                                  |                                                                        | 199.32                        |
| 4. Creating a phone format       | PROC FORMAT;
                                           PICTURE PHONEME
                                           1000000-5000000 = '000-99999'
                                           1000000000-9999999999 = |
                                           395) 999-999999 (PREFIX = ');
| (Schultz: 164)                   |                                                                        | 555-9996                     |
|                                  |                                                                        | (690) 555-0697                |
| 5. Making a picture format into  | PROC FORMAT;
                                           PICTURE EARNINGS
                                           LOW-50000 = 'TOO LITTLE' (NOEDIT)
                                           50001-HIGH = 'TOO MUCH' (NOEDIT); |
| a value-labeled format           |                                                                        | TOO LITTLE                   |
|                                  |                                                                        | TOO MUCH                     |

Figure 6: Enhancing Column Heading Appearance Using PROC FORMAT

PROC FORMAT;
VALUE $CITIES
'NY' = 'NEW YORK'
'BA' = 'BOSTON'
'WA' = 'WASHINGTON D.C.';

DATA AAA;INPUT CITY $ STATE $;
CARDS;
NY NY
WA WA
BA MA;
PROC FREQ;TABLES STATE'CITY';
FORMAT CITY $CITIES. STATE $ST;
RUN;

1269
<table>
<thead>
<tr>
<th>STATE</th>
<th>CITY</th>
<th>FREQUENCY</th>
<th>PERCENT</th>
<th>ROW PCT</th>
<th>COL PCT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BOSTON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>33.33</td>
<td>0.00</td>
<td>0.00</td>
<td>33.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>MASSACHUSETTS</td>
<td>WASHINGTON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0.00</td>
<td>33.33</td>
<td>6.67</td>
<td>33.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NEW YORK</td>
<td></td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>33.33</td>
<td>0.00</td>
<td>0.00</td>
<td>33.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>WASHINGTON D.C.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>33.33</td>
<td>0.00</td>
<td>0.00</td>
<td>33.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>1</td>
<td>33.33</td>
<td>33.33</td>
<td>33.33</td>
<td>100.00</td>
</tr>
</tbody>
</table>

1270