```
*****************************************;
*** The initial part of the program will ***;
*** often have some comments.            ***;
*** ---------------------------------- ***;
*** Little boxes and lines are a nice    ***;
*** way to isolate these comments and    ***;
*** make them stand out.                 ***;
*** ---------------------------------- ***;
*** For this box a comma is not needed   ***;
*** after each line, only after the last ***;
*** one.                                 ***;
*****************************************;

*** In the program below note that skipping lines and indenting lines to improve
readability does not affect the program.  Also note that this comment has run on
for several lines because we have not yet placed a semicolon.
    Of course, all SAS statements end with a ;

/*
Several types of comments are possible.
They do not affect the program in any way,
and can be placed just about anywhere.

Comments are also a good way to inactivate parts of your SAS code.
If there are portions that you do now wish to run, but may want later,
Those segments can be turned into comments to keep them from executing.
*/

/* The statement below will clear the SAS log and SAS output windows */
dm'log;clear;output;clear';

/* Options are available to suppress SAS default centering, dates, page
numbering and labels.  The PS options sets page length (lines) and LS sets    the
line size in columns.  */
options nodate nocenter nonumber ps=512 ls=132;

/* ODS statements are a relatively new SAS innovation that can be used to output
selected tables.  The one below will direct ALL output to a HTML file called
mousefeed02.htme */
ODS HTML style=minimal rs=none
body='C:\Geaghan\Current\EXST3201\Fall2005\SAS\MouseFeed02.html' ;

Title1 'Titles can go most anywhere.';
*** Titles are cumulative, so a title2 or title3 can be added to existing title
statements.  A new title1 or title2 removes all previous title1 or title2
statements AND ANY PREVIOUS TITLES WITH HIGHER NUMBERS.;

/* Input of an external file requires a FILENAME statement to specify the
location of that file. The FILENAME statement needs a reference name, the one
below is called mousein.  This is one place where the limit on SAS names is
rather severe.  This name must be no more than 8 characters.  Careful. */
filename mousein 'C:\Geaghan\Current\EXST3201\Datasets\ASCII\CASE0501.csv';
```

```
*** most every SAS program will require creating a dataset.  The exception is
programs run on datasets that have been stored.  The dataset can be named
anything you wish (up to 16 characters). ***;
data MouseFeed;
   length diet $ 6;
/* SAS can figure variable lengths.  However, character variables are held to 16
characters or less unless a length statement is used.  Numeric variables with
decimals should not be put in a length statement, SAS keeps 16 significant
digits.  Integers are stored with 8 bits, but the size of the dataset can be
reduced with length statements.  Length 3 can store an integer from 1 to 8192,
length 4 an integer from 1 to 2,097,152. */

*** The INFILE statement can be omitted if the data is included in the dataset
below a DATALINES statement and no options are desired.  I usually use an INFILE
statement with a CARDS option (indicating the data is included in the program) in
order to include a MISSOVER option.  This prevents SAS from going to the next
line in order to find missing values.  The FIRSTOBS=# option is needed if there
are some comment lines at the top of the dataset.  The DSD and DLM=',' options
are needed for CSV (comma separated value) files such as those used by the
textbook;
   infile mousein missover DSD dlm="," firstobs=2;

*** The INPUT statement specifies the names of the variables in the dataset.
They can be either numeric or alphanumeric (i.e. character) variables, but
character variables require a $ to specify the alphanumeric type of variable. ;
   input Lifetime diet $;

/* The datalines statement indicates that the data follows, or that the DATA step
ends if the data is not part of the program.  If the dataset is included in the
program it should start immediately below this statement without spaces.  A
semicolon should be placed on the line following the data. */
datalines;    /* If included in the program, the data would follow this line */
;             /* This line would follow the data.  */

/* PROC SORT and BY statements.  Data can be sorted for many reasons,  To make
the data more readable, to locate extreme values or in order to group variables
by some category.  Once grouped, SAS procedures can be run separately on each
group (e.g. BY GROUP).
This is illustrate with the means statements below.*/

Title1 'SAS demo: sort and by statements';
Title2 'PROC Means on whole dataset';
Proc means data=mousefeed; var lifetime; run;

Title2 'PROC Means BY DIET';
/* The following two lines cause the data to be sorted in alphabetical order of
the diets, and then by lifetimes within diets.  Once the data is grouped by diet,
the means can be calculated for each diet with the "BY DIET" statement. */
proc sort data=MouseFeed; by diet lifetime; run;
Proc means data=mousefeed; by diet; var lifetime; run;

*** Note that the diets (the sort BY variable) are sorted into alphabetical order
in a sort statement.  ;
```

```
*** In PROC MEANS the default output values are n, mean, stddev, min and max.
Other variables can be requested including CLM, RANGE, CSS, SKEW, CV, KURTOSIS,
STDERR, LCLM, SUM, SUMWGT, UCLM, USS, VAR, NMISS, the quantiles MEDIAN|P50,
Q3|P75, P1, P90, P5, P95, P10, P99, Q1|P25, QRANGE.  ;

/* Another useful feature of many SAS procs (i.e. procedures) is the ability to
output data. This will be illustrated with the PROC MEANS just used.  The
statements below request a NOPRINT option and add an output statement.  The
output goes to a new SAS dataset that is named by the output.  Since the data is
already sorted, so an additional PROC SORT is not needed. */
Title3 'Means statement with a noprint option and an output statement';
Proc means data=mousefeed noprint; by diet;
   var lifetime;
   output out=dietmean n=n mean=ybar stddev=std stderr=stderr median=med;
run;

Title3 'Listing of means from an output statement';
proc print data=dietmean; run;
*** The dataset listed above was created with an output statement.  This is a
convenient way to obtain results as a file.  Recall that results can also be
obtained by using the ODS statements.  ;

/* Many SAS procedures can do Analysis of Variance.  The most common ones are
PROC GLM and PROC MIXED.  An example of PROC GLM is given below. PROC GLM can be
used with an output statement, but is not used below. */
Title2 'Analysis of Variance with PROC GLM';
/* An important point about the DATA=MOUSEFEED options on the statement below.
Often, we do not need to specify the dataset name because by default SAS process
the last dataset created when no dataset is specified.  Often that is
appropriate.  However, note that at this time the last dataset created was the
PROC MEANS output, DIETMEAN.  Therefore, we must specify the MOUSEFEED data in
subsequent analyses. */
proc glm data=mousefeed;
   class diet;
   model lifetime = diet;
run;
/* Analysis of Variance of variance requires a group, or categorical variable.
This variable is specified in a CLASS statement.  I a variable is not included in
the CLASS statement it is assumed to be quantitative.
The model statement is of the form MODEL = YVARIABLE = XVARIABLE and/or
GROUPVARIABLE.
   */

/* Analysis of Variance is done with PROC MIXED below.  This procedure does not
use an output statement, but has facilities for outputting some variables. In
PROC MIXED the OUTP=somename option on the model statement will automatically
output a variable called PRED with predicted values and a variable called RESID
with the residuals.  All other variables from the original dataset (MOUSEFEED)
are also included in this dataset.  */
Title2 'Analysis of Variance with PROC MIXED';
proc mixed data=mousefeed;
   class diet;
   model lifetime = diet / outp=resids;
* Placing SAS options after a slash is common in many types of SAS statements.;
run;
```

```
Title3 'Listing of PROC MIXED residuals';
proc print data=resids; run;

Title3 'Proc univariate check of the residuals from ANOVA';
proc univariate data=resids plot normal;
   var resid;
run;

/* The original options statement specified a page size of 512 lines, about 10
pages, which I would usually pull into MSWord for processing into separate pages
if necessary.  This page size is obviously too large for graphics.  Therefore, at
this point I reset the page size to 52 lines, about one page.  The line size can
also be set. A page of constant width font, like courier, at a size of 10 is
about 80 columns.  I often use smaller fonts for graphics, so I set a larger line
size. */
options ps=52 ls=99;
Title3 'Scatterplots of PROC MIXED residuals';
proc plot data=resids;
   plot resid * diet / vref=0;
   plot resid * pred / vref=0;
*** Since residual plots ideally show random scatter about the value zero, it is
convenient to have a reference line at zero.  In the plot statement there is
another example of options following a slash, and vref=0 creates a reference line
at the specified value.  ;
run;
/* RUN statements cause SAS to execute any program information collected up to
the point where the RUN is present.  If a series of PROC statements are run, each
PROC acts as a RUN statement for the previous PROC.  However, the last PROC will
not be executed without a RUN statement.  */

options ps=512 ls=99;
/* Here the page size is reset to a large value for any added PROCs. */


QUIT;
/* Some SAS procedures, such as GLM, REG and PLOT remain active after a run
statement.  If you are running SAS interactively and do a plot, it is possible to
add additional plot.  In REG additional MODEL statements can be added, and in GLM
statements like requests for treatment means and contrasts (to be discussed
later) can be added.  In these cases SAS will give the message "PROC GLM running"
in the program window title even though it appears GLM has finished.  A QUIT
statement causes SAS to cease storing information for additional added
statements. */
```

*

In spite of all the comments, this program is an executable SAS program. The color scheme shows comments in green, statement names in blue, PROC and DATA statements in purple, raw data in yellow (not included in this program).

One of the most insidious errors is caused by a missing quote. Quotes for SAS TITLE and LABEL character strings must occur in pairs. A single unpaired quote often results in the message "WARNING: The quoted string currently being processed has become more than 262 characters long. You may have unbalanced quotation marks.". Furthermore, since SAS is "interactive" submitting the program after correcting the problem will not necessarily solve the problem. A second, lone quote must be submitted to undo the first.

;