```
/*
  Welcome,
  In this third introduction you will get to know some advanced and powerful functions in
  Gess Q.

  After you worked through this script you will be able to:
  - copy labels
  - place several questions in a screen
  - use asserts
  - use macros
  - use groups in text replacement and restricts
*/



// - import basic settings -
#include "formats.q"
// - use graphic buttons -
#include "gbuttons.q"




// ---------------------------------------------------------------
// ---------- Survey Contents ------------------------------
// ---------------------------------------------------------------
/*
  Let's begin...

  with a short welcome-text, which is supposed to vary depending on the sex:
*/

singleq question1;
text="Please name your sex.";
labels=
1 "male"
2 "female"
;

HTML{ backButton = yes; };



group welcome;
labels=
1 "Dear Mister X ..." (question1 eq 1)
2 "Dear Misses Y ..." (question1 eq 2)
;

textq question2;
text="
  {welcome}<br>
  We are very pleased, that you are taking part on this Tutorial.
";

/*
  For that a "group" is getting used. A group is not a type of question but a text- and
  structure element. Like types of questions it contains labels followed by constraints in
  brackets. The constraint can be stated as a logical expression (as with filters).
```

```
  Here the group is used in a simple way. In curly brackets they conditionally describe a
  text. One of the labels for that has to have a true logical expression.

  Slightly in advance: normally it is mainly the master for restrict-commands.
*/



/*
  Let's carry on...

  ... with a MultiQ, which contains 20 labels. With the clicked label afterwards it is meant
  to carry on working with. With the command 'maxLabelsPerCol' the labels get distributed
  over 2 columns.
  For fun we randomize the first 10 labels.

  In Tutorial 2 the label rows get repeated and filtered each or altogether with the
  restrict-command. The more elegant way:
*/

HTML{ maxLabelsPerCol = 10; };

multiq question3;
text="Which of the following products do you know?";
title="several statements possible.";
labels=
1  "Product 1 " random
2  "Product 2 " random
3  "Product 3 " random
4  "Product 4 " random
5  "Product 5 " random
6  "Product 6 " random
7  "Product 7 " random
8  "Product 8 " random
9  "Product 9 " random
10 "Product 10" random
11 "Product 11"
12 "Product 12"
13 "Product 13"
14 "Product 14"
15 "Product 15"
16 "Product 16"
17 "Product 17"
18 "Product 18"
19 "Product 19"
20 "Product 20"
;
assert (num(question3) ge 5) "You have to know at least 5 products to be able to carry on.";
// assert (num(question3) ge 5) "You have to know at least 5 products, ..." exit 200;



/*
  New at this point is assert, which has been attached to 'question3'. It checks a certain
  constraint after the click on forward.

  The keyword 'assert' is followed by a logical expression (like with filters).
  Afterwards a default text and an optional abort-code is named (see the commented
```

```
  assert-row).
  If there is an abort-code, by breaching the constraint the interview ends at this point
  with the named code. If there is no code it stays with the actual question and names a
  default text.

  In the example it is getting checked if at least 5 products are known.

  Now we carry on working with the known products:
*/


singleq question4;
text="Which of those products do you use the most?";
labels copy question3;
restrict=question3;

/*
  Instead of the labels=... -listing with "labels copy QNAME;" the labels of a different
  question including their parameters can be taken over.
  Together with the known restrict two of the products chosen beforehand can be picked up
  again in context of a SingleQ.

  Similar things (and even more) can be realized with macros.
  These in the following are only getting basically introduced.
  Macros primarily are text replacement for similar script parts and can save a lot of
  typewrite work.
  But use them with care. Too many macros can lead to bad legibility.

  Almost 5 (quite) similar questions about 5 different products can be asked:
*/

#macro macro_master
singleq question$1;
text="
  Did you buy <span style='color:#$3;'>$2</span> before?
";
labels=
1 "Yes"
2 "No"
;
#endmacro

/*
  A macro with the name 'macro_master' has been defined.
  Up to this point it is only a master. It hasn't got any effect yet.
  Between #macro and #endmacro is the question that we wanted to reproduce several times. It
  is a normal Q. Script – apart from one small fact:

  Dollar symbol followed by a number $1 $2 $3

  This is a dummy variable, of which as many can be built into the code as desired.

  Now we'll instantiate this master with a proper SingleQ:
*/

&macro_master;$1="5";$2="product A";$3="990000";
```

```
/*
  With an AND sign & followed by the macro name now a proper instance of the master is built:
  - &macro_master;
  For that the dummy variables $1 $2 $3 do not have to stay in the script code a replacement
  each has to be named.
  - $1="5"
  replaces the dummy holder with a 5 etc.

  The proper built script code looks the following:

  singleq question5;
  text="
    Did you buy <span style='color:#990000;'>product A</span> before?
  ";
  labels=
  1 "Yes"
  2 "No"
  ;

  Now 5 more instances for the products B,C,D,E with different font colours (HEX statement
  for $3) are built and the relevant questions question5 - question9 are combined in a block:
*/

&macro_master;$1="6";$2="product B";$3="009900";
&macro_master;$1="7";$2="product C";$3="000099";
&macro_master;$1="8";$2="product D";$3="999900";
&macro_master;$1="9";$2="product E";$3="009999";

block macro_block1 = ( question5 question6 question7 question8 question9 );

/*
  At which point and to what extent macros are getting used is up to you. A product list
  from question3 can be built with just one macro too:
*/

#macro macro_labels_question3
1  "product 1 " $1 $2
2  "product 2 " $1    $3
3  "product 3 " $1 $2
4  "product 4 " $1    $3
5  "product 5 " $1 $2
6  "product 6 " $1    $3
7  "product 7 " $1 $2
8  "product 8 " $1    $3
9  "product 9 " $1 $2
10 "product 10" $1    $3
11 "product 11"    $2
12 "product 12"       $3
13 "product 13"    $2
14 "product 14"       $3
15 "product 15"    $2
16 "product 16"       $3
17 "product 17"    $2
18 "product 18"       $3
19 "product 19"    $2
20 "product 20"       $3
#endmacro
```

```
/* Apart from that this list has been addressed with 3 parameters:
 * $1 allows to stop the randomization of the first 10 questions
 * $2 allows to filter the products 1,3,5,7...
 * $3 allows to filter the products 2,4,6,8...
 *
 * But first question 3 is supposed to identically reproduced:
*/

multiq question3_2;
text="Which of the following products do you know?";
title="several answers possible.";
labels=
&macro_labels_question3;$1="random";$2="";$3="";
;

/*
  The randomization is built with the parameter $1="random"; the other parameters are not
  needed and left empty.

  Like this for example label lists can be looked after in a separate file at repeating
  surveys about macros.

  As well with the macro macro_labels_question3 groups can be defined with which certain
  product subgroups can be filtered with.
  2 groups are getting built which can filter products with uneven/even product numbers can
  be filtered:
*/

group grp_products_uneven;
labels=
&macro_labels_question3;$1="";$2="(true)";$3="(false)";
;
group grp_products_even;
labels=
&macro_labels_question3;$1="";$2="(false)";$3="(true)";
;

/*
  In this case the parameter $1 is not needed and left empty,$2="(true)" and $3="(false)" in
  group grp_products_uneven filters the 'uneven products', and round the other way group
  grp_products_even filters the 'even products'. The principle is identical with the group
  used for the welcome text.

  These groups now can be used as parameters for the restrict-command. The randomization
  with $1 in macro_labels_question3 is once active and once left out.
*/

multiq question3_2_uneven;
text="Which of the following products do you know?";
title="several statements possible.";
labels=
&macro_labels_question3;$1="";$2="";$3="";
; restrict = grp_products_uneven;


multiq question3_2_even;
```

```
text="Which of the following products do you know?";
title="several statements possible.";
labels=
&macro_labels_question3;$1="random";$2="";$3="";
; restrict = grp_products_even;

block macro_block2 = (question3_2 question3_2_uneven question3_2_even);


block main = ( question1 question2 question3 question4 macro_block1 macro_block2 );
```