

Generating, administering and marking Dehnadi-style tests

Richard Bornat, Saeed Dehnadi
School of Science and Technology, Middlesex University, London, UK
R.Bornat@mdx.ac.uk, S.Dehnadi@mdx.ac.uk

November 4, 2013

We have automated the business of generating a questionnaire which includes the questions of Dehnadi's test, and other similar tests. We have also automated the analysis of responses to the test. This document describes and explains what you have to do to use the tests and our generation / analysis tool.

Contents

1	How to get the tests, the tool, and our papers	4
2	Basics	4
2.1	Nomenclature: questionnaire, test, survey	4
2.2	Generating, administering and marking questionnaires	4
2.3	Running the generator/analyser program	4
3	Generating a questionnaire	5
3.1	What kind of questionnaire?	6
3.1.1	Can you afford to use SurveyMonkey?	6
3.1.2	Could you use LimeSurvey?	6
3.1.3	Could you use paper?	6
4	Activating an online questionnaire	6
4.1	Activating a LimeSurvey questionnaire	6
4.2	Activating a SurveyMonkey questionnaire	7
5	Gathering results	7
5.1	Gathering results from LimeSurvey	7
5.2	Gathering results from SurveyMonkey	8
5.3	Gathering results from a paper test	8
6	Analysing the results	8
7	Understanding the analysis	9
7.1	Correct responses	9
7.2	The judgements	9
7.3	Subdivisions of Algorithmic	10
7.4	Particular non-algorithmic responses	10
7.5	Assignment models	10
7.6	Sequence models	11
A	Description files	12
A.1	What to do if you can't edit the description files	12
B	Questionnaire description	13
B.1	Command line	13
B.2	Command arguments	13
B.3	HTML text	13
B.4	Compulsory commands	13

B.5	Named directives and options	13
B.6	Compulsory questions	14
B.7	Conditional execution – **If , **Else , **Elsf , **Fi	14
B.8	Commands	14
B.8.1	**ArrayChoice	14
B.8.2	**Authorisation	14
B.8.3	**Authorization	15
B.8.4	**Choice	15
B.8.5	**Else	15
B.8.6	**Elsf	15
B.8.7	**Fi	15
B.8.8	**Final	15
B.8.9	**Goodbye	15
B.8.10	**Group	16
B.8.11	**If	16
B.8.12	**MultiChoice	16
B.8.13	**MultiText	16
B.8.14	**Number	16
B.8.15	**Option	16
B.8.16	**Questionnaire	17
B.8.17	**Title	17
B.8.18	**Text	17
B.8.19	**Welcome	17
C	Questionnaire description	18
C.0.20	**Id	18
C.0.21	**Title	18
C.0.22	**Questions	18

1 How to get the tests, the tool, and our papers

All that you need (including this manual) is available from [our distribution page](#) at Middlesex University.

Some of our papers are available from [Saeed Dehnadi's page](#); the latest is [available from PPIG](#).

If you want to administer our paper test, download it from [here](#) and follow the instructions in section ?? and section ??.

If you want to generate a questionnaire, or analyse the responses to a questionnaire, download our generation / analysis program and the files it needs from [our distribution page](#).

If you want to see what an online test looks like, see [our sample questionnaire](#).

If you're in a school or university we may be able to host an online test for you: contact [Richard Bornat](#).

If you want to understand the analyser's judgements or the models of assignment and sequence that we use, see section 7.

2 Basics

2.1 Nomenclature: questionnaire, test, survey

When Saeed Dehnadi was carrying out his research, he talked of administering a 'test' to his students, or sometimes a 'questionnaire'. When we began to automate his work, we found that the software we wanted to use talked about 'surveys'. We found also that it was natural to divide the questions that we asked our test participants into two kinds:

personal questions about their background (e.g. what is your name? have you ever written a computer program?);

programming questions about assignment and sequence (e.g. `int a=10; int b=20; a=b;` what are the new values of a and b?)

In this document we have called the whole thing – personal and programming questions together – a *questionnaire*, and the part of it that is about programming a *test*, but LimeSurvey and SurveyMonkey will continue to call the questionnaire a 'survey'.

2.2 Generating, administering and marking questionnaires

We have written a program which generates questionnaires as files that can be loaded into a survey tool – LimeSurvey or SurveyMonkey – or printed out and administered as a paper test. LimeSurvey and SurveyMonkey will administer the test and output a spreadsheet detailing all the responses; we distribute an empty spreadsheet with the paper version which you can fill in yourself from the questionnaire scripts. In each case our program can then analyse the responses in the spreadsheet.

To use this manual you have to decide whether you are going to generate a questionnaire (section 3), administer a questionnaire (section ??), analyse the responses to a questionnaire (section 6) or understand the analysed responses (section 7).

2.3 Running the generator/analyser program

To generate or analyse a questionnaire you have to use our generator/analyser program. It is written in Java and distributed as an 'executable jar' file called `DehnadiGenAn.jar`. Before you can use it you may

need to install Java on your computer, if you don't have it already.

On Windows or OS X, you double-click the `jar` file and it runs. On Linux don't double-click (it unpacks the file instead of running it): `java -jar DehnadiGenAn.jar` will run it properly.

3 Generating a questionnaire

Start the program (section 2.3). It asks you whether you want to generate a questionnaire (choose Generate), and then asks to open two files:

- a questionnaire description file;
- a test description file.

The questionnaire description file describes all the personal information and background questions; the test description file (separate because one test fits many different questionnaires) describes the programming questions.

In case you want to generate your own questionnaire and/or test descriptions, see appendix ?? . But you will probably use one of the files we distribute, or we might produce a questionnaire description just for you.

In any case you will probably use a standard test description file. We distribute two:

- Saeed's original test (`ClassicDehnadiTest.txt`);
- a modified version of the original test (`NonRepeatingDehnadiTest.txt`).

The original test is provided for historical reasons, but we have found a problem with it: many of the answers to the last six questions are shared, and a participant who repeatedly ticks the same answer will seem to be using the same models of assignment and sequence throughout, which could be misleading.¹ So ***we strongly recommend that you use the modified test*** from the file `NonRepeatingDehnadiTest.txt`.

Once it has its questionnaire and test description files, the program will ask what kind of questionnaire you want to generate: the choice at present is between LimeSurvey, SurveyMonkey and Paper: see section 3.1 if you aren't sure what kind to choose. Once you have chosen it will generate the questionnaire files, plus copies of the questionnaire description and test description, which will be needed when the responses are analysed. Each of the files it generates will have as part of its name a *questionnaire id*, a pseudo-random sequence of eight characters which is unique to the questionnaire you generated.

- A *LimeSurvey* questionnaire is a single file called `questionnaire_⟨questionnaire id⟩.lss`, plus `QD_LimeSurvey_⟨questionnaire id⟩.txt` and `TD_LimeSurvey_⟨questionnaire id⟩.txt`. Keep all three files. See section 4.1 for what to do with the `.lss` file, and section ?? for what to do with the QD and TD files.
- A *SurveyMonkey* questionnaire is a single file called `questionnaire_⟨questionnaire id⟩.txt`, plus `QD_SurveyMonkey_⟨questionnaire id⟩.txt` and `TD_SurveyMonkey_⟨questionnaire id⟩.txt`. Keep all three files. See section 4.2 for what to do with the questionnaire file, and section ?? for what to do with the QD and TD files.
- A *Paper* questionnaire is a pair of \LaTeX files called `questionnaire_⟨questionnaire id⟩.tex` and `answersheet_⟨questionnaire id⟩.tex`. Run these through \LaTeX to generate postscript or pdf

¹ But it might not be. Those participants who we have informally interviewed after they have done this seem to know what they are doing, so we haven't lost faith in Saeed's original results.

or dvi or whatever printable format you prefer. You also get `QD_Paper_{questionnaire id}.txt`, `TD_Paper_{questionnaire id}.txt` and `responses_{questionnaire id}.csv`: keep all three files, and see section ?? for what to do with them.

3.1 What kind of questionnaire?

There are advantages and disadvantages of each of the questionnaire types.

3.1.1 Can you afford to use SurveyMonkey?

The bald fact is that *you must pay for a SurveyMonkey account to make it useful*. The free (Basic) account will only take ten questions (Saeed's test has twelve, and you will need more to gather at least the names of your participants) and, crucially, it won't allow you to download the responses for analysis.

3.1.2 Could you use LimeSurvey?

LimeSurvey is geeky to setup, and there are some security issues (get techie assistance), but it's free. We have a LimeSurvey setup at Middlesex, and if you are in a school or university, we may be able to host your survey: contact us.

3.1.3 Could you use paper?

A paper survey is free, once you've printed enough questionnaires, though it's tedious transferring all the responses to a spreadsheet. Once you've done that we've automated the process: see section ??.

4 Activating an online questionnaire

At present you have to use LimeSurvey (free, see section 3.1.2) or SurveyMonkey (paid, because the free Basic account is too basic – see section 3.1.1).

4.1 Activating a LimeSurvey questionnaire

Login to your LimeSurvey website, press the big green '+' sign that lets you import or create a new survey, and point it at the `.lss` file you generated (section 3). It should load without incident. You may like to 'integrity check' it (inside the pencil, called 'survey properties', there is a QA button, labelled 'survey logic file').

Test the survey with the green gearwheel icon.

If you don't like the survey title, change it in 'survey properties': 'edit text elements'.

Once you are happy with the survey, activate it by pressing the green triangle next to the red square. Activation is reversible (don't panic!).

If you want the activated survey to appear on the front page of the website, you can go to 'survey properties': 'general settings': 'publication and access control' and check 'list survey publicly'. At the same time you need to set a start date and an expiry date.

Most of the other settings have been made at the time the `.lss` file was generated. If you find a useful one that wasn't set, please let us know.

Give participants the link to your survey, or the link to the website front page if the survey is available from there.

4.2 Activating a SurveyMonkey questionnaire

You will need a paid SurveyMonkey account: the free SurveyMonkey service won't deal with more than ten questions, and the test part alone of your questionnaire contains twelve questions.

You *must paste the text* from the `questionnaire_<questionnaire id>.txt` file into the SurveyMonkey website, because SurveyMonkey incorporates it into the spreadsheet of responses which it generates, and the automated analysis process recognises that text in order to be able to interpret the responses.

At present SurveyMonkey can deal with all the question types (see appendix ??) that the questionnaire generator can process. Here is what you must do with the text from the `questionnaire_<questionnaire id>.txt` file. (*This section under development, because I'm finding it difficult to get to a paid SurveyMonkey account myself ...*)

****Text** : corresponds to SurveyMonkey 'Single Textbox' with question text copied from the .txt file;

****Number** : corresponds to SurveyMonkey 'Numerical Textboxes' with question text copied from the .txt file, and a single answer label of your choice;

****MultiText** : corresponds to SurveyMonkey 'Multiple Textbox' with question text copied from the .txt file, and answer fields copied likewise;

****Choice** : corresponds to SurveyMonkey 'Multiple Choice (only one Answer)' with question text copied from the .txt file, and answer fields copied likewise;

****MultiChoice** : corresponds to SurveyMonkey 'Multiple Choice (Multiple Answers)' with question text copied from the .txt file, and answer fields copied likewise (but if there is an 'Other' field, you may want to allow text input for that);

****MultiText** : corresponds to SurveyMonkey 'Multiple Textboxes', with question text copied from the .txt file, and answer fields copied likewise

The programming questions are dealt with as 'Multiple Choice (Multiple Answers)'. Paste the question text (from 'Read the following statements' up to 'The new value of ...'). Paste the answers (a list of lines 'a=...', 'b=...', ...) into the Answer Choices. Tick 'add Other', paste 'any other values ...' into the Field Label. Don't require an answer. Click 'Randomise, flip or sort choices' and choose 'Randomize answer choices for each respondent'.

5 Gathering results

5.1 Gathering results from LimeSurvey

Login to the website, choose the survey from the dropdown list of surveys, choose the icon that looks like a speech bubble behind a pie chart, called 'Responses'; within that choose 'Responses and statistics'. The icon which looks like a cylinder with a solid green arrow, called 'export results to application', leads you to the output page.

Choose the responses you want (range 1 to n, completion state). Under Headings click ‘Question code’ (and don’t click ‘convert spaces’); under Responses choose ‘answer codes’ (and don’t click ‘convert Y’ or ‘convert N’); under Format click ‘CSV file’. Then press ‘Export data’.

It will produce a file called results-surveyNNNNN.csv, where NNNNN is the internal number (chosen by the website) of your survey. Your browser puts the file in the place where it puts downloads.

5.2 Gathering results from SurveyMonkey

(This section under development, because I’m finding it difficult to get to a paid SurveyMonkey account myself ...)

5.3 Gathering results from a paper test

Paper tests were originally designed to be marked by hand, using Saeed’s marking sheet. But that’s tedious and error prone. If you must use a paper test, it can still be marked automatically. The `responses_{questionnaire id}.csv` file produced when the questionnaire was generated is a spreadsheet which contains headers that identify the tickboxes for each question, in the order they were placed on the question sheet.

Only use the responses spreadsheet with the questionnaire for which it was generated: the tickboxes are randomly ordered, different every time a questionnaire is generated.

First load the `responses_{questionnaire id}.csv` file into a spreadsheet program. It will show you a spreadsheet with columns in which you can record the results of personal questions, as well as columns for the test questions. Don’t delete any of the columns: the analyser looks at the column headers to make sure it is dealing with responses to the right questionnaire.

Each row shows a single participant’s responses. Each question gets several columns: those labelled “Qn: `<answer>[m]`” correspond to tickboxes from the questionnaire for question *n*; those labelled “Qn: Other” are for responses from the ‘any other values’ slots for question *n*. The tickboxes are listed in the order they appear in the printed questionnaire, so it’s easy to put a mark (non-blank, something like Y or *) to correspond to each tick on the questionnaire script. In the ‘Other’ column you can write the additional answers, if any, from the ‘any other values’ answers: put commas between the parts of the answer (e.g. a=20, b=7), and semicolons between answers, if the participant has given more than one.

Save the spreadsheet as a csv file – don’t let your spreadsheet program bully you into choosing a more specialised format.

6 Analysing the results

The process for marking a questionnaire is the same for all types. You run the generator/analyser program (section 2.3); choose Mark rather than Generate; identify the responses file (a csv file produced as in section 5), and the questionnaire description and test description used to generate the questionnaire – i.e. the QD and TD files produced as in section 3. From the QD and TD filenames the program can see how you gathered the results, and analyse the responses accordingly.²

If you don’t give the program the right files it will complain, and in most cases it won’t then be able to do its analysis. Contact Richard Bornat if you have difficulties.

² If you lose those files then it is still possible to analyse the results, if you can identify the questionnaire and test descriptions that were used to generate the questionnaire. But it’s not so easy, so please don’t lose the files. And **definitely** don’t lose the responses file generated for a paper test.

The program will output its analysis in a `csv` file that you can load into a spreadsheet program, and a diagnostic `txt` file giving an explanation of how it made its judgements. The `csv` file is probably more interesting to you: it is the same as the responses, with some additional columns called 'rating', 'reason', and Q1, Q2, etc. which show the answer given to each question and the model(s) which that answer matches in Dehnadi's interpretation.

7 Understanding the analysis

There are three ways in which we know that a Dehnadi test can be used:

1. pre-course, to see how students view programming questions;
2. during a course, to test student progress and see which of them need help;
3. after a course, to test student achievement see what effect the teaching had.

In the first case you will be interested in whether students appear to be using an *algorithmic* mental model or not; those who don't may struggle at the start of the course. In the second case you will be interested to see whether students have the *correct model* already; those who seem to be using an incorrect model and those who are still not using any model at all will need extra help. The third case is like the second, but you will be more interested in your overall success rate.

In the analysis spreadsheet, the column headed *judgement* contains the program's judgement of each response. The column *reason* gives the mental models that seem to have been used and the number of times they have been observed. The columns labelled Q1, Q2 etc. give the program's classification of the answer(s) to each question.

7.1 Correct responses

In a pre-teaching test you will be looking for responses labelled Algorithmic or Possibly algorithmic, and not bothering too much about the particular models that were used. In the second and third cases you will be looking for overall *correct responses*: **Algorithmic (overall)** in the judgement column and **M2+S1** in the reason column.

7.2 The judgements

When he devised his test, Saeed was looking for students who consistently – i.e. throughout the test – used an algorithmic model of assignment. He labelled their responses 'consistent', and others 'inconsistent'. But the words he chose read like psychometric measurements of the students rather than their responses, so we've renamed them.

Broadly we divide responses into three groups:

Blank responses have fewer than 8 answers;

Algorithmic responses appear to use the same mental model(s) on many answers;

Unrecognised responses are those which are neither Blank nor Algorithmic.

We don't see many Blanks. The majority of responses, even in our experiment with 14-year-old schoolchildren, are Algorithmic – usually two thirds, and sometimes many more. We chose Unrecognised rather than Inconsistent to indicate that those responses are simply not understood; they may be rational and even algorithmic in ways that we haven't anticipated.

7.3 Subdivisions of Algorithmic

In his thesis Saeed recognised as algorithmic those responses which used the same models in 8 out of 12 questions *or* which used the same models in the first 6 questions. We've added those which use the same models in 6 out of the last 9 (the multi-variable questions).

Saeed's marking algorithm concentrated on the assignment model (section ??) and ignored the sequence model (section 7.6). The analysis tool is stricter: it classifies as *Possibly algorithmic* those responses which use the same assignment but not always the same sequence model.

That gives us six judgements:

Algorithmic (overall) – same models in 8 of 12 answers;

Algorithmic (first 6) – same models in first 6 answers;

Algorithmic (last 9) – same models in 6 of the last 9 answers;

Possibly algorithmic (overall) – same assignment model in 8 of 12 answers;

Possibly algorithmic (first 6) – same assignment model in first 6 answers;

Possibly algorithmic (last 9) – same assignment model in 6 of the last 9 answers;

A large majority of Algorithmic judgements are Algorithmic (overall).

We've also added some experimental judgements, but we aren't confident that the tool can accurately identify the relevant responses, so don't put too much faith in them.

Sequentially algorithmic responses appear to use one model for a while, and then switch to another;

Concurrently algorithmic responses use more than one model at once, giving an answer to each question for each model.

Both these kinds of response are rare, and concurrently algorithmic responses – even if we can recognise them accurately – are very rare indeed.

7.4 Particular non-algorithmic responses

In our experiment with schoolchildren a few participants ticked every box. That seems to us to be a sort of protest response, and we have a judgement (**Ticked Everything**) for it.

One of Saeed's assignment models (M9 in section ??) is No Change. The respondent ticks the box which lists the original values of the variables, a rational answer which indicates that the assignment statement in the question has no effect. It's rational, but it's not algorithmic, so it has a judgement (**No change**) of its own.

7.5 Assignment models

section ?? See section 1 for Saeed's models of the Java/C assignment $a=b$.

M1-M8 and M11 are recognisable assignment models. M9 is a non-algorithmic model. M10 seems to arise from an interpretation of the equality sign as algebraic equality.

Note that M10 answers require as many ticks as there are variables – in two-variable questions two as shown in section 1 – in three-variable questions three ticks: $a=A, b=A, c=A; a=B, b=B, c=B; a=C, b=C, c=C$ – and so on).

The 'correct' model is of course M2.

Table 1: Models of assignment $a=b$ where initially $a=A, b=B$

M1	Right-to-left move	$a=B, b=0$
M2	Right-to-left copy	$a=B, b=B$
M3	Left-to-right move	$a=0, b=A$
M4	Left-to-right copy	$a=A, b=A$
M5	Right-to-left copy-and-add	$a=A+B, b=B$
M6	Right-to-left move-and-add	$a=A+B, b=0$
M7	Left-to-right copy-and-add	$a=A, b=B+A$
M8	Left-to-right move-and-add	$a=0, b=B+A$
M9	No Change	$a=A, b=B$
M10	Equality	$a=A, b=B$ and $a=B, b=B$
M11	Swap	$a=B, b=A$

Table 2: Models of composition $C1 \ C2$

S1	Sequential	give the result of applying C2 to (the result of applying C1 to the initial state).
S2	Parallel	give the result of applying C1 to the initial state and the result of applying C2 to the initial state.
S3	Parallel destination-only	use ‘copy’ in place of ‘move’ (i.e. M2 rather than M1, M4 rather than M3, etc.), and then as S2.

7.6 Sequence models

See section 2 for Saeed’s model of composition $C1 \ C2$. (Note that in Java and C the semicolon is part of the assignment statement ... Isn’t it wonderful?)

S3 is quite a common choice, and it’s the source of the ambiguity about assignment models which makes marking the test so complicated.

A Description files

A questionnaire falls naturally into two parts: questions about the participant themselves; and the programming questions that investigate the way they think about programming. These two parts are described by separate text files.

A.1 What to do if you can't edit the description files

Mac OS X and Linux use a line-feed character (LF) to show the end of a line. Windows uses two characters – carriage return and line feed (CR and LF). In the Bad Old Days, before OS X, Mac used carriage-return (CR).

If you can't edit the description files then it's probably because you are running Windows and the files use LF rather than CRLF. You could complain to the person who sent you the files, or you could use a program (e.g. EditPad) that can convert them for you.

B Questionnaire description

The file contains a sequence of blocks. Most blocks describe questions (like ‘what is your name?’) or texts (like ‘hello, welcome to the programming questionnaire’). Blocks will be displayed to questionnaire participants in the order they are given, though there are ways of skipping some blocks (see `**If`).

B.1 Command line

The first line of a block is always a command line. A command line starts with two asterisks followed immediately (no space allowed) by a command word like `Title` or `Authorisation`. If there is more stuff on the command line the command word is followed by a space and then that extra stuff. What follows the command line depends on what command it is. Here, for example, is how you might describe a question which asks a person’s gender:

```
**Choice Gender 2 true
Gender
**Option M
Male
**Option F
Female
```

See below for more information about `**Choice` blocks.

B.2 Command arguments

Most command lines include arguments to the command – a space separated list of items. In the example above the question is named ‘Gender’, it includes two options, and it is compulsory. The first option is called M and the second option is called F. Note that the block contains three sub-blocks: the question text (here a single word) and two options, each of which contains option text (here, each a single word).

B.3 HTML text

All text which is to be displayed to the participant – e.g. ‘Gender’, ‘Male’ and ‘Female’ in the example above – is interpreted as HTML. Most text-editing programs let you insert HTML styling markers like `<i>` and `</i>`. Good luck.

B.4 Compulsory commands

You must include at least a questionnaire title (`**Title`) and a questionnaire (`**Questionnaire`). Different questionnaire types may require more: in a LimeSurvey questionnaire, for example, you must also include `welcome` (`**Welcome`) and `goodbye` (`**Goodbye`) commands.

B.5 Named directives and options

Online survey software wants named questions and options. So all the questions and options have a ‘name’ argument.

B.6 Compulsory questions

You may wish to force a participant to answer a question: you might really need to know their student number, for example. The last argument of a command is normally ‘true’ or ‘false’: true for compulsory, false for optional.

B.7 Conditional execution – ****If**, ****Else**, ****Elsf**, ****Fi**

You may have a question which you want to ask only if the participant has answered some earlier question – usually a ****Choice** – in a particular way. You insert a ****If** directive

```
**If <qname> <optname>
```

and what follows, up to the next ****Else**, ****Elsf** or ****Fi** (whichever comes first) will be used only if the participant chose option *optname* as the answer to question *qname*.

****Else** works as you would expect: what follows is shown in case the preceding ****If** fails. ****Elsf** <*qname1*> <*optname1*> replaces ****Else** followed by ****If**, and reduces the number of ****Fi** command lines you need.

For each ****If** there must be exactly one ****Fi**, a closing bracket to the conditional.

B.8 Commands

B.8.1 ****ArrayChoice**

Shows an array of radio buttons. The participant may pick one radio button on each line.

```
**ArrayChoice <name> <compulsory>  
<text>  
<columns>  
<rows>
```

Each *column* is

```
**Vertical <name>  
<text>
```

– the column label is *text*. Each *row*, similarly, is

```
**Horizontal <name>  
<text>
```

B.8.2 ****Authorisation**

Used when you want to ask the participant’s permission to do something with the questionnaire results. Gives them the alternative of accepting or refusing. Abandons the questionnaire if they refuse.

```
**Authorisation <name>  
<text1>  
**OK <text2>  
**NotOK <text3>  
<text4>
```

The participant sees *text1* with two radio buttons labelled *text2* and *text3*. If they select *text3* the next page they see shows *text4* and that, for that participant, is the end of the questionnaire.

B.8.3 **Authorization

Same as ****Authorisation**, for people who want to spell like Americans (or Oxfordians).

B.8.4 **Choice

Shows a set of radio buttons from which the participant must choose one. See also ****MultiChoice**.

```
**Choice <name> <numoptions> <compulsory>  
    <text>  
    <options>
```

For options, see ****Option**; there must be exactly *numoptions* of them.

B.8.5 **Else

See section [B.7](#).

B.8.6 **Elsf

See section [B.7](#).

B.8.7 **Fi

See section [B.7](#).

B.8.8 **Final

Comes at the end of the questionnaire, just before ****Goodbye** (if included). Gives a large text box for participants to past comments in.

```
**Goodbye  
    <text>
```

B.8.9 **Goodbye

A page of text to be shown after all other questions (unless authorisation is refused, when the last page is *text4* from the authorisation question).

```
**Goodbye  
    <text>
```

B.8.10 ****Group**

Essentially, start a new page of questions

```
**Group <groupname> <text>
```

text is the title of the new page.

B.8.11 ****If**

See section [B.7](#).

B.8.12 ****MultiChoice**

Shows a set of tick boxes which the participant may tick. If compulsory, at least one must be ticked. See also ****Choice**.

```
**MultiChoice <name> <numoptions> <compulsory>  
<text>  
<options>
```

For options, see ****Option**; there must be exactly *numoptions* of them.

B.8.13 ****MultiText**

A question with several sub-questions, to each of which the user may type some text. (May, not must: even if the question is compulsory, not every sub-question need be answered. I think.)

```
**MultiText <name> <numoptions> <compulsory>  
<text>  
<options>
```

For options, see ****Option**; there must be exactly *numoptions* of them (why? belt and braces? I dunno).

B.8.14 ****Number**

A special version of ****Text** in which the response which the participant types must be a numeral.

```
**Number <name> <compulsory>  
<text>
```

B.8.15 ****Option**

An alternative or a sub-question.

```
**Option <name>  
<text>
```


B.8.16 **Questionnaire

Insert the questions of the programming questionnaire.

****Questionnaire**

The questions of that questionnaire are inserted into the questionnaire at this point. Usually you will put ****Questionnaire** after the personal questions and before ****Final** and ****Goodbye** (if included).

B.8.17 **Title

Gives the title you choose for your questionnaire.

****Title** *<text>*

B.8.18 **Text

Ask a question to which the participant must type some text.

****Text** *<name>* *<compulsory>*
<text>

Displays a text-input box labelled with *text*.

B.8.19 **Welcome

A page of text to be shown before any other questions (even authorisation questions).

****Welcome**
<text>

C Questionnaire description

Questionnaire descriptions are much simpler than questionnaire descriptions. They must start with a ****Id**, they may contain a ****Title**, and they must contain ****Questions**. And that's it.

C.0.20 ****Id**

Must come first in the questionnaire description.

Gives the identifier of the questionnaire. Sort of version control.

```
**Id <text>
```

C.0.21 ****Title**

Gives a title for the questionnaire. I've forgotten what happens if you put a title in both the questionnaire description and the test description. Perhaps it shows both.

```
**Title <text>
```

C.0.22 ****Questions**

The meat of the questionnaire.

```
**Questions  
<questions>
```

Each question is on a single line; blank lines are ignored. It consists of some Java/C declarations followed by some assignments. For example:

```
int a = 5; int b = 3; int c = 7; a = c; b = a; c = b;
```

The generator complains if the assignments use undeclared variables, or if there is a multiple declaration of a single variable name. Otherwise (I think) anything goes.