Tutorial: Spec-based Testing

1. You are asked to test a module, which calculates the cost of buying cans of beer from Beer Store. The basic cost per can is $10, but discounts are available for bulk orders. If you order between 10 and 99 cans you qualify for a 10% discount, while if you order 100 cans or more you qualify for a 20% discount. The module should take as input the required number of cans, and output the price of the order.

   Use EP to test this module.

2. Your task is to comprehensively test the **studentgroups,** a program which maintains a list of student records.

   ## System Description of StudentGroups

   The system is to maintain a list of students. Each item in the list will hold the student's name, version of degree and group. To maintain the list the system must be able to add to the current list and remove from the list. An individual student will be uniquely identifiable by name.
   The system must be able to show a list of all students for a specified group.

   For each student the following information will be stored:

name	any 40 characters
version	0 non-language
	1 French
	2 German
group	any single uppercase character

   The following options must be available at menu 1:

M	maintain (i.e. add or remove) student records
A	list all students
G	list by group
Q	quit the program

   If G is selected above, then the user must be prompted for the group.

   If M is selected the following options must be offered at menu 2:

A	add student
R	remove student
Q	quit menu 2 and return to menu 1

   If R is selected above then, after entering the student's name, the user must be asked to confirm the request to delete.

   The data will be stored on a file of fixed length records, one per student.

Normally new student records will be added to the end of the file. However, if a student record is deleted then that fixed length slot will be reused if a new student record is added.

The filename is to be entered as a system parameter when the program is executed.

Design spec-based test cases. These test cases must be of two types, as follows.

1) Verifying that valid data is handled correctly

Tests must be designed to verify that the program handles valid input correctly.

Examples of valid input include the following fields for each student - a name of up to 40 characters, a version (0 for non-language students, 1 for Computing with French, 2 for Computing with German) and a group (any uppercase letter).

Normal operation of the program consists of the following two operations:-

- maintaining student records, i.e. adding or removing them
- listing student records, either all of them or a specified group.

The program will create a new list or maintain an existing one.

Design test cases to exercise all the possible operations on all types of *valid* student records on the list. Use the Test Cases sheets to record your planned test cases. Leave the right hand column blank for your test results.

2) Trying to break the program

The second type of tests is to be designed with the aim of trying to break the program. Think of the most destructive, abusive, crazy things you can do to the program and add them to your test cases.

Test Cases			**Tester:**	**Date:**
Test Case No.	Operation Tested	Input	Expected Output or Effect	Test Run Results

Test Cases			**Tester:**	**Date:**
Test Case No.	Operation Tested	Input	Expected Output or Effect	Test Run Results