

1 Introduction

The C programming course CM100138 is assessed by a sat examination and also an on-line examination. The sat examination is of the traditional kind, taking place in a room; past papers are available. The online examination is in place of coursework and is a kind of assessment you may not have met before. This mock examination emulates the real online examination to give you some experience of the real online exam.

The emulation is as close as possible, but it *necessarily* differs in some important ways which are explained at the bottom of this paper. You may care to take this paper with you read about the differences after your mock online examination.

2 The mock online examination

Your lab tutor will tell you how to download a simple calculator program, called `calc.c` and two associated data file called `test.in` and `test.out`. Do as follows. (1) Download the program and data files into your C workspace. (2) Complete the assignment below. (3) Send your the documents specified in the assignment to the email address given to your by your lab tutor.

Note: Follow the above instructions exactly. If you send anything other than is required, your tutor is not obliged to mark anything. In the real exam marks are given for the ability to obey instructions.

2.1 The assignment

The calculator program was demonstrated in class. It is capable of addition and multiplication. Input is via the keyboard, unless redirected to a file. Output is via the terminal, unless redirected to a file. The calculator processes characters on the input stream as follows

- `n` the next input token is a number which is read and placed at top of the calculator's store.
- `+` removes the the top two number in store, adds them and stores the result at the top.
- `*` removes the the top two number in store, multiplies them and stores the result at the top.
- `q` Quits the program.
- Any other characters on the input stream are ignored.

After each input token is processed, the calculator prints the current value in the top of its store.

The assignment comes in several parts, do each part strictly in order

Debugging syntax errors

1. Include your name and email in your program using a comment as the first line of the program.
2. `calc.c` contains several syntax errors. Correct these errors.
3. Running your code with the input file called `test.in`. This should output identical to `test.out`. Test your corrected code by making sure this happens.

Debugging runtime errors

Some users have complained that the program sometimes give the wrong values. In real industrial situation you would be expected to find out why, and fix the problem. Here you are told it is because the the calculators store has limited size. Rather than fix the problem you must warn users if they try to store too many numbers. The program must issue the warning `store full` if the user tries to exceed the calculators limits, and `too few numbers` if the user tries to multiply or add numbers when the store contains less than two numbers.

4. Emulate the problem by change the store size to have a limit of 4 numbers. You are expected to find out how to do this for yourself. Convince your self you have emulated the problem by running the program using an input file of your own devising called `mytest.in`.

5. Devise a solution and code it into `calc.c`. The manner of solution is up to you, but you are strongly advised to look at lines of code such as `i = newnumber(array, N, i);`. This is poor code for at least two reasons: (i) They change a variable that is important to the operation of the store outside the functions that control the store (these are the library functions). (ii) There is no way for the calling function to recognise whether the library function has worked successfully or not.

6. Test your program by showing that it issues appropriate warnings when run with `mytest.in` as input. Redirect output to `mytest.out`. Further test your code by showing it produces the correct input when the supplied file `test.in` is run. The calculator should not issue any warnings for this input.

Here are some hints: (i) If you use pointers (or structures) then you can solve the first problem in a neat way that open up the possibility of a neat solution to the second solution. (ii) Think very carefully about which part of the program should issue warnings, in particular consider whether library functions should do *more* than originally intended.

2.2 What to submit, what to do next

Change the name of your program `calc.c` to `calc.email.c` where `[email]` is replaced with the first part of your BUCS email address. For example, if your BUCS email is `abc123@bath.ac.uk`, then change you program name to `calc.abc123.c`. Do likewise to your `mytest.in` and `mytest.out` data files.

Email your program and both of your own test data files to the address given by your tutor. You are reminded to submit nothing else.

Your program will be compiled and run on the data files you have submitted. It will also be run on data files held in private by the tutors. You will receive feedback by email. The feedback is formative in nature, it does not contribute to your assessment.

3 Differences from the real online exam

As mentioned, the mock online exam closely emulates the real online exam in that you get to write code under the same conditions. This experience is to be valued, despite an important difference that is now explained.

The real online examination will ask you to write a small application using *libraries* that you have already written as part of your laboratory work. The mock examination cannot assume you have written these libraries, or even that you know about libraries as that part of the course has not been taught when the mock examination is sat.

As a reminder, the real online examination assumes that you have written *libraries* for the abstract data types `Array2D_int`, `Stack_int` and `Btree_int`, which support two dimensional arrays, stacks, and binary trees each containing the `int` data type. Since these will be assessed, you must design and write them yourself.

You will be asked to write an application that uses at least one of these libraries. You will submit for marking the application, the libraries you used (but strictly not those you have not used), and any supporting documentation such as a `Makefile`. The implication is that your libraries have to be generic, and your coding skills improved compared to the level expected in the mock examination.

The real online examination serves several purposes at once:

- It emulates industrial conditions. Programmers in industry are typically given tasks they have not seen before, and are required to use libraries to complete those tasks. The line-managers will more often than not require the task is completed in a given times.
- It tests your coding ability directly. There is no better test on one's ability to code than to be asked to write code. The compiler will not compile incorrect syntax, and a compiled program either produces a correct output, without crashing, or it does not.
- It guards against plagiarism. Some people have submitted beautifully written libraries, but an application so badly written it is not even C. The best explanation is that they have not written the library themselves, a conjecture supported when the same library is found to be submitted by another. This may be an extreme case, but even a difference in programming style is enough to warn an experienced C programmer that the library and application may have different authors.