**Software Engineering 265**
**Software Development Methods**
**Spring 2013**

*Assignment 4*

Due: Wednesday, April 3rd, 9:00 am by submission via Subversion
(no late submissions accepted)

## Programming environment

For this assignment you must ensure your work executes correctly on the Linux machines in ELW B215. You are free to do some of your programming on your own computers; if you do this, give yourself a few days before the due date to iron out any bugs in your C solution that might appear once it is used on BSEng machines in ELW B215.

## Individual work

This assignment is to be completed by each individual student (i.e., no group work). Naturally you will want to discuss aspects of the problem with fellow students, and such discussion is encouraged. **However, sharing of code fragments is strictly forbidden without the express written permission of the course instructor (Zastre).** If you are still unsure regarding what is permitted or have other questions about what constitutes appropriate collaboration, please contact me as soon as possible. (Code-similarity analysis tools may be used to examine submitted programs.)

## Objectives of this assignment

- Use the dynamic-memory support available in C (i.e., "malloc()").
- Use some of the separable-compilation features of C.
- Use a "makefile".
- Perform some error-handling.
- Use Subversion to manage changes in your source code and annotate the evolution of your solution with "messages" provided during commits.
- Test your code against the twenty provided test cases.
- Be prepared to justify your script's handling of errors during the assignment demonstration.

**This assignment: "s265fmt3.c" & "uvic_formatter.c"**

At this point you are probably a little tired of seeing the formatting problem. However, on the bright side you thoroughly understand what output is expected for the given inputs. Now you can concentrate on what is perhaps the most difficult aspect of programming in C: dynamic memory. For this assignment you will be wrapping a solution to the formatting problem a C file / module. The module provides different access functions depending on whether a file's contents are to be formatted or an array of strings. The formatted output will be returned from the called function as a dynamically-allocated array of strings.

- s265fmt3.c will accept a filename as an argument and will call the appropriate routine in uvic_formatter. The strings in the returned string array will be output to stdout.

- If no filename is provided to s265fmt3, then the contents of stdin will be formatted.

- driver.c already creates a dynamically-allocated array of strings and passes this to the appropriate routine in uvic_formatter. As with s265fmt3.c, the array of strings returned from the routine are to be output to stdout.

- The specifications from the second assignment are to be used for this assignment. However, the interpretation to be used for "?mrgn" commands must be that of the files in /home/zastre/seng265/assign4.

- Several files have already been provided for you in /home/zastre/seng265/assign4/_code. You are to use these files when starting your work. Ensure they are placed in the "assign4" directory of your project.

- Ensure there is some error handling.

- With your completed "s265fmt3" program, the input would be transformed into the output (here redirected to a file) via one of the two following UNIX commands:

```
% ./s265fmt3 /home/zastre/seng265/assign4/in11.txt > ./myout11.txt

% cat ~zastre/seng265/assign4/in11.txt |./s265fmt3 > ./myout11.txt
```

where the file "myout11.txt" would be placed in your current directory.

**Exercises for this assignment**

1. Within your Subversion project create an "assign4" subdirectory. Use the test files in /home/zastre/seng265/assign4. (Files in01.txt through to in20.txt are the same as those used for the second assignment.) Your completed "s265fmt3.c" and "uvic_formatter.c" files (plus "driver.c" and "uvic_formatter.h") must appear here. Ensure the subdirectory and files are added to Subversion control.

2. You are free to use elements of the instructor's solution to assignment #1 as part of your work. (This code is available in the assignment #4 directory.) However, please note my A#1 solution does not use dynamic memory, and the code may not necessarily work with formatting extensions (i.e., relative margins) covered in assignment #2.

3. **Your solution may not assume a maximum length for an input line, nor may you assume a maximum number of lines.**

4. Reasonable run-time performance of your program is expected. None of the test cases should take longer than 15 seconds to complete on a machine in ELW B215.

**What you must submit**

- Three C files named "s265fmt3.c", "driver.c" and "uvic_formatter.c" within your subversion repository containing a solution to Assignment #4.

- If you have modified the "makefile" or "uvic_formatter.h", then ensure these files are also in the subversion repository and be prepared to justify at demo time why you needed to change these files. (An e-mail obtained the instructor ahead of time okaying such changes and shown at the demo can act as this justification.)

- A text file named "error_handling.txt" which enumerates the errors that are handled by your submission.

**Evaluation**

Students will demonstrate their work to a member of the course's teaching team. Sign-up sheets for demos will be provided a few days before the due-date; each demo will require from 10 to 15 minutes.

Our grading scheme is relatively simple.

- "A" grade: An exceptional submission demonstrating creativity and initiative. "s265fmt3" and hence the code in "uvic_formatter.c" run without any problems. "driver.c" runs without any problems. Errors have been enumerated and are suitably handled. The program is clearly written and uses functions appropriately (i.e., is well structured).
- "B" grade: A submission completing the requirements of the assignment. "s265fmt3" and hence the code in "uvic_formatter.c" run with few problems. "driver.c" runs without few problems. Errors have been enumerated and are suitably handled. The program is clearly written.
- "C" grade: A submission completing most of the requirements of the assignment. "s265fmt3" and hence the code in "uvic_formatter.c" run with some significant problems. "driver.c" might or might not run without any problems. Some important error cases have been overlooked or are not handled.
- "D" grade: A serious attempt at completing requirements for the assignment. "s265fmt3" and hence "uvic_formatter.c" run with several serious problems.
- "F" grade: Either no submission given, or submission represents very little work.