

COMP10002 – Assignment 1

1	The Task
2	Submission Process
3	Today's Work & Advices

The Task in a nutshell

Input: A text file (like `alice-eg.txt`):

```
Down, down, down. Would the fall NEVER come to an end! 'I wonder how
many miles I've fallen by this time?' she said aloud. 'I must be getting
```

```
...
```

Command to build: `./ass1 lat long < alice-eg.txt`

Output:

```
S1: query = lat long
```

```
---
```

```
Down, down, down. Would the fall NEVER come to an end! 'I wonder how
```

```
S2: line = 1, bytes = 68, words = 14
```

```
S3: line = 1, score = 0.000
```

```
---
```

```
...
```

```
-----
```

```
S4: line 9, score = 0.668
```

```
or Longitude I've got to?' (Alice had no idea what Latitude was, or
```

```
...
```

(see `test0-out.txt` for full output)

The Task: Input

Input: `argc`, `argv[]` and a text file.

Things should consider about the input text file and storing inputs:

- text file format (is a sequence of chars, or words, or strings?)
- what limits (file size, or length of words, or length of strings)
- how to read and keep inputs? (pay attention on the note that at any moment you are not allowed to keep all strings in memory)

The Task: Stage 1 with incremental development

Stage 1: processing command line.

Approach:

- First, process arguments and print out, and, when suitable, print out error message and terminate program.
- Test that that works.
- Output of Stage 1 should be:

```
$ ./ass1 < alice-eg.txt
```

```
S1: No query specified, must provide at least one word
```

```
$ ./ass1 lat 66 loNg 32 words < alice-eg.txt
```

```
S1: query = lat 66 loNg 32 words
```

```
S1: loNg: invalid character(s) in query
```

```
$ ./ass1 lat long < alice-eg.txt
```

```
S1: query = lat long
```

```
---
```

The Task: Stage 2 with incremental development

Stage 2: reading and printing input, line by line.

Approach:

- Probably need to read and output each line of input.
- Output of Stage 2 should be:

```
$ ./ass1 ali lat long < alice-eg.txt
```

```
S1: query = ali lat long
```

```
---
```

```
Down, down, down. Would the fall NEVER come to an end! 'I wonder how
```

```
S2: line = 1, bytes = 68, words = 14
```

```
---
```

```
many miles I've fallen by this time?' she said aloud. 'I must be getting
```

```
S2: line = 2, bytes = 72, words = 15
```

```
---
```

```
somewhere near the centre of the earth. Let me see: that would be four
```

```
S2: line = 3, bytes = 70, words = 14
```

```
---
```

```
...
```

The Task: Stage 3 with incremental development

Stage 3: with each line, compute and print the similarity score. Need to think how to compute score, how to check if a string is a prefix of another string, ignoring case (perhaps think first, then ask Mr Google?)

- Output of Stage 3 should be:

```
$ ./ass1 ali lat long < alice-eg.txt
S1: query = ali lat long
---
Down, down, down. Would the fall NEVER come to an end! 'I wonder how
S2: line = 1, bytes = 68, words = 14
S3: line = 1, score = 0.000
---
many miles I've fallen by this time?' she said aloud. 'I must be getting
S2: line = 2, bytes = 72, words = 15
S3: line = 2, score = 0.000
---
somewhere near the centre of the earth. Let me see: that would be four
S2: line = 3, bytes = 70, words = 14
S3: line = 3, score = 0.000
---
thousand miles down, I think--' (for, you see, Alice had learnt several
S2: line = 4, bytes = 71, words = 12
S3: line = 4, score = 0.229
---
```

The Task: Stage 4 with incremental development

Stage 4: Need to keep track of (up to) five lines that have highest similarity scores. How, what data structure should be used?

- Output of Stage 4 should be:

```
$ ./ass1 ali lat long < alice-eg.txt
```

```
...
```

```
-----
```

```
S4: line = 9, score = 0.668
```

```
or Longitude I've got to?' (Alice had no idea what Latitude was, or
```

```
---
```

```
S4: line = 10, score = 0.233
```

```
Longitude either, but thought they were nice grand words to say.)
```

```
---
```

```
S4: line = 4, score = 0.229
```

```
thousand miles down, I think--' (for, you see, Alice had learnt several
```

```
---
```

```
S4: line = 8, score = 0.226
```

```
'--yes, that's about the right distance--but then I wonder what Latitude
```

```
---
```

?

Assignment 1: CCTS process

1. **CREATE**: Create a directory, say `ass1`, download all related files into `ass1`, then create `ass1/ass1.c` that satisfies the requirements 😊
2. **COPY**: Copy the whole directory `ass1` to your university's drive `H:`. Note: if you work in lab computers and use `H:`, you don't need to do this step.
3. **TEST**: login into the server `dimefox.eng.unimelb.edu.au`, then on that server, navigate to the directory `ass1`, compile and test your program.
4. **SUBMIT**: while in `dimefox`, submit your `ass1.c`, and verify.

Today Work

Create a simple (perhaps empty, perhaps just for Stage 1), then try all 4 steps. Make sure that you can submit, at least from a lab PC.

Then, incrementally **CREATE** your `ass1.c`, do **COPY-TEST-SUBMIT** after every major development.

1. The CREATE step (on lab PCs or your laptop)

CREATE: Create an assignment's directory, say `ass1`, under your `comp10002`. To this directory:

- download all the data files mentioned in point 2 of FAQ, namely, `alice-eg.txt`, `pg11.txt`, download all `data*.txt` and `test*-out.txt`
- then create near-empty `ass1/ass1.c`, compile & test to make sure it “works”.

2. The COPY step (from your laptop)

COPY: Copy the whole directory `ass1` to your university's drive `H:`.

1. If you use your laptop/desktop at home: you need to install `VPN` for remote access to uni's computers. See Alistair's [Submission instructions](#) from [FAQ](#) for how to.

2. To copy:

- If yours is a Mac: open a `Terminal`. If it's a PCs: open a `minGW` window [if you don't have `minGW`, install it or alternatively install `pscp` and `putty` as told by [Submission Instructions](#)]
- Navigate to the parent directory of your `ass1`
- Run the following command for copying the whole directory `ass1`:

```
scp -r ass1 XXX@dimefox.eng.unimelb.edu.au:
```

(note: replace `XXX` with your `loginname`, and don't forget the `colon` at the end of the line; if you use `pscp`, then use that instead of `scp`)

3. The TEST step (supposing that you've ass1.c working)

- **login into the server** `dimefox.eng.unimelb.edu.au`: From Mac `Terminal`, or Windows' `MinGW` window, run command:

```
ssh dimefox.eng.unimelb.edu.au
```

- **Then**, when you are with `dimefox`:
 - Navigate to your `ass1` directory
 - Compile your program
 - Test, at least with all data Alistair supplied.
- **Example testing** using `alice-eg.txt` and `test0-out.txt`:

```
$ ./ass1 ali lat long < alice-eg.txt >mytest0-out.txt
```

```
$ diff mytest0-out.txt test0-out.txt
```

The “`diff`” command will find the difference between 2 files. If it produces no output at all, then the 2 files are absolutely identical (Bravo!). If not, then you need to open both files using `jEdit` and try to figure out what's wrong in your output.

4. The submit process

When you are working on `dimefox`, and already navigated to your `ass1` directory, run:

```
submit comp10002 ass1 myass1.c
```

then, wait a few minutes and verify by:

```
verify comp10002 ass1 > my-receipt-ass1.txt  
more my-receipt-ass1.txt
```

The “`more`” command will display the content of the receipt. Alternatively, you can use `jEdit` to open `my-receipt-ass1.txt` for a careful viewing.

When to submit: Submit now, submit today, submit after any session you work with the assignment. Think about submission as a way to backup your work!

Assignments: advices

- *Be active in the subject's Discussion Forum!*
- *Make as many submissions as you want, only the last one (before deadline) counts. Deadline: **10:00AM on Mon 18 September!***
- *To simplify, do submit at uni. If you want to submit from home, then **install VPN today!***
- *Read the specifications carefully.*
- *Test your program carefully, at least with all supplied data. Do the testing not only in your computer, but also on dimefox.*
- ***Read the marking rubric carefully and try to maximize your marks!***
- ***START EARLY, AIM TO FINISH EARLY!***