COMP10002 – Assignment 1

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

The Task in a nutshell

Input: A text file such as **`mytest.txt`** :

```
ccccccccccccccccccatg
gaaaaaaaaaaaaaaaaatt
ttttcccccccccccccccc
```

Command: **`./ass1 < mytest.txt > mytest-out.txt`**

Output file **`mytest-out.txt`**:

```
Stage 0 Output
--------------
3 fragments read, 60 characters in total

Stage 1 Output
--------------
 0: frg= 0, slen= 20  Ccccccccccccccccccatg
 1: frg= 1, slen= 39  CccccccccccccccccccatGaaaaaaaaaaaaaaaaatt
 2: frg= 2, slen= 57  CccccccccccccccccccatGaaaaa .. aaaaaTtttcccccccccccccccccc
---
 2: frg=-1, slen= 57  CccccccccccccccccccatGaaaaa .. aaaaaTtttcccccccccccccccccc

Stage 2 Output
--------------
```

The Task: Input

Input: from `stdin`.

Things should consider about the input text:

- maximal number of lines = ?
- maximal number of chars per line = ?
- do we need, and how to, store all input?
  - during processing, do we need to add anything to each input?
  - what data structure for keeping inputs?
- how to read each input line:
  - can we simply use `scanf(…)` ?
  - do we really need `getchar()` or `mygetchar()` ?

Incremental development:

- Read the marking rubric
- Start with an empty program that meets the marking rubrics
- Implement Stage 0, and test it with a couple of data files
  - In DEBUG mode, print out all input data
  - Check: marking rubric is followed well,
  - Check: output for stage 0 is correct,
  - Check: DEBUG printouts are the same as input data
- Then, implement Stage 1, and test it with all 3 data sets:
  - What data structure for output? Do we need to keep it after finishing Stage 1?
  - Always think about a major operation as a separate function
  - Perhaps the function will be employed further? How to make it generous for that purpose?
  - Check: correctness, marking rubric

?

Computing environment in your laptop: scp, ssh, VPN

- Make sure that your `Unix`/`minGW` shell have commands `scp` and `ssh`
- If your `minGW` does not accept `scp` and `ssh`, open the `minGW Installation Manager`, mark line "`msys_openssh bin`" for installation and install.
- At home: install `VPN` for connecting to uni's server from home by following the link from Submission Instructions

- Sample copying a file or a folder to **`H:`**

```
scp ass1.c bob@dimefox.eng.unimelb.edu.au:
scp –r ass1 bob@dimefox.eng.unimelb.edu.au:
```

- Login into `dimefox` /`nutmeg`

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

Assignment 1: 4C 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. `CHECK`: login into the server to do the testing/checking `dimefox.eng.unimelb.edu.au`, then on that server, navigate to the directory `ass1`, compile and test your program.
4. `COMMIT`: while in `dimefox`, submit your `ass1.c`, and verify.

Today Work
Create a simple (perhaps near 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**-**CHECK**-**COMMIT** 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, all three files `test?.txt` and three files `test?-out.txt`
- then create near-empty `ass1/ass1.c`, by:
  - copying the content of `ass1-skel.c` , and
  - perhaps add comment `Algorithms are fun` at the end
- then 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.

2. To copy:
   - If yours is a Mac: open a Terminal. If it's a PCs: open a minGW window
   - Navigate to the parent directory of your ass1
   - Run the following command for copying the whole directory ass1:

```
scp –r ass1 bob@dimefox.eng.unimelb.edu.au:
```

(note: replace bob with your loginname, and don't forget the colon : at the end of the line)

3. The CHECK (TEST) step

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

  ```
  ssh bob@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 `test0-out.txt`:

```
$ ./ass1 < test0.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.

You can also do testing on your laptop, but remember that a final test in `dimefox` is a need!

4. The COMMIT (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, scp and 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 17 September!***
- *If you want to submit from home, then* ***install VPN today****!*
- *Read the specifications and marking rubric carefully.*
- *Test your program carefully, at least with all supplied data. Do the testing not only in your computer, but also on dimefox.*
- *Invent some more data files (especially for some extreme cases) for testing.*

- ***Read the marking rubric carefully and try to maximize your marks!***
- ***START EARLY, AIM TO FINISH EARLY!***