COMP10002 – Assignment 1

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

Assignment: The Task

The Task: Stage 1 in a nutshell

Purpose: Write a program, say, `ass1`.c and compile to `ass1`.

Input: from `stdin` - a text file such as `test0.txt`

Output: to `stdout`

Sample Interface:

```
./ass1 < test0.txt > my_test0_out.txt
```

Minimal Requirements:

- Obeying all programming rules we learnt so far,
- Following well the marking rubric,
- Optional use of `struct` ,
- NOT using `malloc`,
- Including a mandatory authorship declaration at the start of the program,
- output being exactly as the corresponding supplied sample output, ie. `my_test0_out.txt` being identical to `test0_out.txt`

Submission

Submission:

- is done by using a remote computer (`dimefox` or `nutmeg`), and
- could be problematic sometimes.

➔ Do a submission today
to avoid any painful technical problem later

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, say ass1.c, to the root directory of **H:**

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

- Login into dimefox /nutmeg

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

Incremental development:

- Start with a near-empty program by downloading from the link provided in section 3 of FAQ, then signing the declaration.
- Copy to H: (if you are on your laptop), and submit

- Then, implement Stage 1, and test it with all 4 data sets:
  - Always think about a major operation as a separate function.
  - After adding a major operation, check and make sure that the program works as expected.
  - Check: correctness, marking rubric

?

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, 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 the directory `ass1`, under your `comp10002`.

```
cd comp10002
mkdir ass1
cd ass1
```

To this directory:

- download all the data files supplied in section 2 of FAQ , namely, all four files `test?.txt` and four files `test?-out.txt`
- then create near-empty `ass1/ass1.c`, by:
  - *downloading* the content of `ass1-skel.c` , sign, and
  - perhaps add comment `Algorithms are fun` at the end
- then compile & test to make sure that this simplest program works.

2: The COPY step (from your laptop)

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

To copy:

- Mac: open a Terminal. PC: open a minGW window
- Navigate to the parent directory of your ass1
- Run the 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)

Note: From next time, you only need to copy ass1.c. Use:

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

3: The CHECK (=testing on dimefox) step [NOT FOR TODAY]

- **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

login into the server `dimefox` **if not yet done:** From Mac `Terminal`, or Windows' `MinGW` window, run command:

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

Then, on `dimefox`, run:

```
cd ~/ass1
submit comp10002 ass1 myass1.c
```

wait a bit then verify your submission by:

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

When to submit? Submit now, submit today, `scp` and `submit` after any session you work with the assignment.

Assignment 1: What needs to be done today?

1. make a directory (say, `ass1`) for the assignment
2. copy all data files into `ass1`
3. build `ass1`.c from `ass1-skel.c`
4. compile and test
5. copy `ass1` to uni's `H:` if needed
6. compile and test if needed
7. `submit`
8. `verify`
9. go back to program, implement Stage 1, goto step 4

NOTE: 15 minutes before end_of_class, do steps 5, 7, 8 regardless of the success of other steps.

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 23 September!***
- *If you want to submit from home, then* ***install VPN today****!*
- *Read & follow 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!***