COMP10002 Workshop Week 11 (for self study only)

Outlook:

1	Working in your assignment 2 (a gift from Alistair!)
2	Learn to use gdb and valgrind (if having time or having troubles with your codes)

gdb for debugging

Sometimes, `gdb` can be very helpful in debugging programs. You should spend time to try `gdb` if:

- You have already finished your assignment. In this case, make a copy of your `ass2` and "invent" some errors, and see how `gdb` can help.

  (OR check your program with `valgrind` first, it might still have some problems)
- You haven't finished your assignment, and you believe that your algorithm is correct, but still have some weird errors.

gdb

Learning gdb is easy, and you can do it yourself. Just take a good tutorial from the Internet, by googling “gdb tutorial”.
Or, you can use 2 .pdf files attached in our github.

REMEMBER: to be able to use gdb (and valgrind) you should compile your program with the flag `–g`
Normal compilation:

```
gcc –Wall –o ass2 ass2.c
```

Compilation for using with `gdb`/`valgrind`:

```
gcc –g –Wall –o ass2 ass2.c
```

running gdb on dimefox

If you are lucky enough to have `gdb` on your laptop, then just use your laptop.

If you have to use `dimefox`, you'd better to employ a lab's PC, because you need 2 windows side by side:

- One for `jEdit` so that you can view and make changes to your program.
- One for `ssh dimefox`, and then compiling, using `gdb`, `valgrind` etc.

Now, after having 2windows opened, you can start to do the tutorial by following `gdb-tutorial-handout.pdf`.

Later on (not now), you might find that `GDB_cheetset.pdf` is useful.

valgrind – checking for memory leaks

Compile your program with flag –g

Now, suppose that you execute your program with:

```
./ass2 b <test1.txt
```

To run valgrind, just prepend the word “valgrind” before your command:

```
valgrind ./ass2 b <test1.txt
```

You will see a valrind report mixed with your output. If you just wnt to see valgrind report you can use, for eample:

```
valgrind ./ass2 b <test1.txt >myoutp.txt
```

A perfect valgrind report should have the content similar to:

```
==9755== Memcheck, a memory error detector
==9755== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
==9755== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
==9755== Command: ./h
==9755==
hhhh= 5
==9755==
==9755== HEAP SUMMARY:
==9755==     in use at exit: 0 bytes in 0 blocks
==9755==   total heap usage: 1 allocs, 1 frees, 4 bytes allocated
==9755==
==9755== All heap blocks were freed -- no leaks are possible
==9755==
==9755== For counts of detected and suppressed errors, rerun with: -v
==9755== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 8 from 6)
-b
```

One trouble some program has report ended with:

```
==9684== HEAP SUMMARY:
==9684==     in use at exit: 736 bytes in 31 blocks
==9684==   total heap usage: 32 allocs, 1 frees, 864 bytes allocated
==9684==
==9684== LEAK SUMMARY:
==9684==    definitely lost: 352 bytes in 7 blocks
==9684==    indirectly lost: 384 bytes in 24 blocks
==9684==      possibly lost: 0 bytes in 0 blocks
==9684==    still reachable: 0 bytes in 0 blocks
==9684==         suppressed: 0 bytes in 0 blocks
==9684== Rerun with --leak-check=full to see details of leaked memory
==9684==
==9684== For counts of detected and suppressed errors, rerun with: -v
==9684== ERROR SUMMARY: 24 errors from 6 contexts (suppressed: 8 from
6)
-
```

valgrind

`valgrind` can also give some other information. If you want to go more with that, ask our friend `Google`.

Next week

As part of the next week's program, we will cover number representation. I hope that after that you won't need to review that part for the exam ☺