Operating Systems 2018-B
Assignment 1: Shell and Command Line

Deadline: Monday, 19/03/2018

This assignment should be submitted as a *.tar.gz* archive, with script files task1, task2, etc. inside. The files in the archive must have correct permissions (e.g., the executable bits).

The assignment should be submitted in groups of 2 students. Names and IDs of the students in the group must be written inside a README file in the archive, and *also* listed in the submission comment in Moodle.

Here is how you can create a flat submission archive of directory ex1 that contains files ex1/README, ex1/task1, ex1/task2, and so forth:

```
tar -C ex1 -czvf ex1.tar.gz .
```

Questions about the assignment should be asked in Piazza. Late submissions are penalized 10 points per day, and assignments can be submitted at most 5 days late. Special requests should be directed to your lab TA.

Task 1

Some people use the first line of every text file as a heading or description of the file's contents. A report that lists the filenames and first line of each text file can make sifting through several hundred text files a lot easier.

Write a single line command that lists the first line in every text file in your home directory in a report file (report.txt).

Task 2

Write a Bash script that uses `chmod` and `find` commands to change permissions of all files and directories in the current directory as follows: rwxr-x--- for files with .sh extension, rw-r----- for files without an .sh extension, and rwx--x--- for directories. Symbolic links and other special files should be ignored.

Use `man find` and `man chmod` to look up suitable switches. See especially the `-type` and `-exec` switches of `find`.

Note that when testing the script, it cannot be assumed to reside in current directory. E.g.: `nano task1; mkdir test1; cd test1; […]; ../task1`

Task 3

Write a Bash script that uses *for* loops and conditionals in order to recursively copy the current directory into ../backup directory, which needs to be created if it does not exist.

See documentation for Bash *for* loops, and for `-f` and `-d` switches of the `[` command (used for conditionals). E.g., the command `for i in *; do echo $i; done` prints all filenames in the current directory.

You should not use the `-r` switch of `cp` command. However, you can assume that current directory depth is limited to 2 (directories don't contain other directories), all filenames are simple (no spaces or other special characters), and there are no hidden files (filenames beginning with a dot).

Task 4

Write a Bash script that creates a dictionary for a file that is given as a parameter. E.g., if the script is run as `./task3 mypoem`, it should create the file `mypoem.dict`, containing all unique lowercased alphabetic words of length 3–8 in file mypoem, one per line.

The script parameter can be accessed using `$1` variable. You can build a single complex command, which connects simple commands using "`|`" pipes, and optionally uses output redirects. See the manual of `grep` (especially `-E` and `-o` switches) for locating alphabetic words using regular expressions, manual of `tr` for lowercasing words, and manual of `sort` for sorting without repetitions. Check out what `grep -Eo '\<[a-c1-3]{2,3}\>'` does.

Task 5

Write a Bash script that uses gcc in a loop to compile all .c source files in a current directory into .o object files, and additionally creates .s assembly listings in Intel format. Then, the .o files are linked into runnable executable. The object files may be compiled from the assembly listings using the as assembler, or directly compiled from .c sources as it is usually done.

The files should be compiled with size optimizations, all warnings enabled, 2011 ANSI C syntax, and debugging information intact. See `man gcc` for the relevant options (use "/" to search the manual). Make sure to check out the following options: `-c`, `-o`, `-S`, `-masm`, `-std`, `-W`, `-g`.