COMP[29]041 16s2 (http://www.cse.unsw.edu.au/~cs2041/16s2/)

Simple Shell Scripts

Software Construction (http://www.cse.unsw.edu.au/~cs2041/16s2/)

Aims

This exercise aims to give you practice with using the Unix shell for processing collections of files.

Assessment

Submission: `give cs2041 lab03 digits.sh echon.sh file_sizes.sh` also submit `courses.sh` if you attempt the challenge exercises

Deadline: either during the lab, Monday 15 August 11:59pm (midnight)

Assessment: Make sure that you are familiar with the lab assessment criteria (lab/assessment.html).

Exercise 1: Mapping Digits

Write a program `digits.sh` that reads from standard input and writes to standard output mapping all digit characters whose values are less than 5 into the character ' < ' and all digit characters whose values are greater than 5 into the character ' > '. The digit character '5' should be left unchanged.

Sample Input Data	Corresponding Output
`1 234 5 678 9`	`< <<< 5 >>> >`

`I can think of 100's`
`of other things I'd rather`
`be doing than these 3 questions`
	`I can think of <<<'s`
	`of other things I'd rather`
	`be doing than these < questions`

`A line with lots of numbers:`
`12345678912345678912345678 9`
`A line with all zeroes`
`00000000000000000000000000`
`A line with blanks at the end`
`1 2 3`
	`A line with lots of numbers:`
	`<<<<5>>>>>>>><5>>>>><<<<5>>>>`
	`A line with all zeroes`
	`<<<<<<<<<<<<<<<<<<<<<<<<<<`
	`A line with blanks at the end`
	`< < <`

`Input with absolutely 0 digits in it` `Well ... apart from that one ...`	`Input with absolutely < digits in it` `Well ... apart from that one ...`
`1 2 4 8 16 32 64 128 256 512 1024` `2048 4096 8192 16384 32768 65536`	`< < < > <> << >< <<> <5> 5<< <<<<` `<<<> <<>> ><>< <><>< <<>>> >55<>`

You can run some tests on your script like this:

```
$ ~cs2041/bin/autotest lab03 digits.sh
```

Also do your own testing!

Exercise 2: Repeated Echo

Write a shell script a program `echon.sh` which given exactly two arguments, an integer *n* and a string, prints the string *n* times. For example:

```
$ ./echon.sh 5 hello
hello
hello
hello
hello
hello
$ ./echon.sh 0 nothing
$ ./echon.sh 1 goodbye
goodbye
```

Your script should print exactly the error message below if it is not given exactly 2 arguments:

```
$ ./echon.sh
Usage: ./echon.sh <number of lines> <string>
$ ./echon.sh 1 2 3
Usage: ./echon.sh <number of lines> <string>
```

Also get your script to print this error message if its first argument isn't a non-negative integer:

```
$ ./echon.sh hello world
./echon.sh: argument 1 must be a non-negative integer
$ ./echon.sh -42 lines
./echon.sh: argument 1 must be a non-negative integer
```

Although its better practice to print your error messages to `stderr` print your error messages to stdout for this exercise.

Hint: you'll need to use the shell `if`, `while` and `exit` statements, shell arithmetic and the `test` command.

You can run some tests on your script like this:

```
$ ~cs2041/bin/autotest lab03 echon.sh
```

Also do your own testing!

Exercise 3: Files Sizes

Write a shell script `file_sizes.sh` which prints the names of the files in the current directory splitting them into three categories: *small*, *medium-sized* and *large*. A file is considered *small* if it contains less than 10 lines, *medium-sized* if contains less than 100 lines, otherwise it is considered *large*.

Your script should always print exactly three lines of output. Files should be listed in alphabetic order on each line. Your shell-script should match character-for-character the output shown in the example below. Notice the creation of a separate direcory for testing and the use of the script from the last question to produce test files. You could also produce test files manually using an editor.

```
$ mkdir test
$ cd test
$ ../echon.sh 5 text >a
$ ../echon.sh 505 text >bbb
$ ../echon.sh 17 text >cc
$ ../echon.sh 10 text >d
$ ../echon.sh 1000 text >e
$ ../echon.sh 0 text >empty
$ ls -l
total 24
-rw-r--r-- 1 andrewt andrewt   25 Mar 24 10:37 a
-rw-r--r-- 1 andrewt andrewt 2525 Mar 24 10:37 bbb
-rw-r--r-- 1 andrewt andrewt   85 Mar 24 10:37 cc
-rw-r--r-- 1 andrewt andrewt   50 Mar 24 10:37 d
-rw-r--r-- 1 andrewt andrewt 5000 Mar 24 10:37 e
-rw-r--r-- 1 andrewt andrewt    0 Mar 24 10:37 empty
$ ../file_sizes.sh
Small files: a empty
Medium-sized files: cc d
Large files: bbb e
$ rm cc d
$ ../echon.sh 10000 . >lots_of_dots
$ ls -l
total 36
-rw-r--r-- 1 andrewt andrewt    25 Mar 24 10:37 a
-rw-r--r-- 1 andrewt andrewt  2525 Mar 24 10:37 bbb
-rw-r--r-- 1 andrewt andrewt  5000 Mar 24 10:37 e
-rw-r--r-- 1 andrewt andrewt     0 Mar 24 10:37 empty
-rw-r--r-- 1 andrewt andrewt 20000 Mar 24 10:39 lots_of_dots
$ ../file_sizes.sh
Small files: a empty
Medium-sized files:
Large files: bbb e lots_of_dots
$
```

Hint: you can use the command `wc` to discover how many lines are in a file. You probably want to use the shell's back quotes, its `if` statement, and the `test` command and

You can run some tests on your script like this:

```
$ ~cs2041/bin/autotest lab03 file_sizes.sh
```

Also do your own testing!

Challenge Exercise: Scraping Courses

Write a shell script `courses.sh` which prints a list of UNSW courses with the given prefix by extracting them from the UNSW handbook webpages. For example:

```
$ courses.sh OPTM
OPTM2111 Optometry 2A
OPTM2190 Introduction to Clinical Optometry
OPTM2211 Optometry 2B
OPTM2291 Primary Care Optometry
OPTM3111 Optometry 3A
OPTM3131 Ocular Disease 3A
OPTM3211 Optometry 3B
OPTM3231 Ocular Disease 3B
OPTM4110 Optometry 4A
OPTM4131 Clinical Optometry 4A
OPTM4151 Ocular Therapeutics 4A
OPTM4211 Optometry 4B
OPTM4231 Clinical Optometry 4B
OPTM4251 Ocular Therapeutics 4B
OPTM4271 Professional Optometry
OPTM4291 Optometry, Medicine & Patient Management
OPTM5111 Clinical Optometry 5A
OPTM5131 Specialist Clinical Optometry 5A
OPTM5151 Clinical Ocular Therapeutics 5A
OPTM5171 Research Project 5A
OPTM5211 Clinical Optometry 5B
OPTM5231 Specialist Clinical Optometry 5B
OPTM5251 Clinical Ocular Therapeutics 5B
OPTM5271 Research Project 5B
OPTM7001 Introduction to Community Eye Health
OPTM7002 Epidemiology & Biostatistics for Needs Assessment
OPTM7003 Epidemiology of Blinding Eye Diseases
OPTM7004 Advocacy and Education in Community Eye Health
OPTM7005 Eye Health Economics and Sustainability
OPTM7006 Eye Care Program Management
OPTM7007 Community Eye Health Project
OPTM7103 Behavioural Optometry 1
OPTM7104 Advanced Contact Lens Studies 1
OPTM7108 Research Skills in Optometry
OPTM7110 Public Health Optometry
OPTM7115 Visual Neuroscience
OPTM7117 Ocular Therapy 2
OPTM7203 Behavioural Optometry 2
OPTM7205 Specialty Contact Lens Studies
OPTM7213 Ocular Therapy
OPTM7301 Advanced Clinical Optometry
OPTM7302 Evidence Based Optometry
OPTM7308 Research Project
OPTM7444 Business Skills in Optometry
OPTM7511 Advanced Ocular Disease 1
OPTM7521 Advanced Ocular Disease 2
$ courses.sh MATH|wc
    126     585    4874
$ courses.sh COMP|grep Soft
COMP2041 Software Construction: Techniques and Tools
COMP3141 Software System Design and Implementation
COMP3431 Robotic Software Architecture
COMP4001 Object-Oriented Software Development
```

```
COMP4161 Advanced Topics in Software Verification
COMP4181 Language-based Software Safety
COMP9041 Software Construction: Techniques and Tools
COMP9181 Language-based Software Safety
COMP9431 Robotic Software Architecture
$ courses.sh MINE|grep Rock
MINE3630 Rock Breakage
MINE8640 Geotechnical Hazards in Hard Rock Mines
MINE8660 Geotechnical Engineering for Underground Hard Rock
```

Your script must download the handbook web pages and extract the information from them when it is run.

Hints

This task can be done using the usual tools of grep, sed, sort & uniq but the regular expressions take some thought.

The UNSW handbook uses seperate web pages for undergraduate and postgraduate courses. These two web pages would need to be downloaded for the above example (JAPN): `http://www.handbook.unsw.edu.au/vbook2016/brCoursesByAtoZ.jsp?StudyLevel=Undergraduate&descr=O` (http://www.handbook.unsw.edu.au/vbook2016/brCoursesByAtoZ.jsp?StudyLevel=Undergraduate&descr=O) and `http://www.handbook.unsw.edu.au/vbook2016/brCoursesByAtoZ.jsp?StudyLevel=Postgraduate&descr=O` (http://www.handbook.unsw.edu.au/vbook2016/brCoursesByAtoZ.jsp?StudyLevel=Postgraduate&descr=O).

Make sure courses which occur in both postgraduate & undergraduate handbooks aren't repeated.

`cat -A` can be useful to check for non-printing characters.

The command `wget` can be used to download a web page and has option which allow it to be used in shell pipelines. For example:

```
$ wget -q -O- "http://www.handbook.unsw.edu.au/vbook2016/brCoursesByAtoZ.jsp?StudyLevel=Underg
  <TD class="" align="left">OPTM2111</TD>
  <TD class=""><A href="http://www.handbook.unsw.edu.au/undergraduate/c
  <TD class="evenTableCell" align="left">OPTM2190</TD>
  <TD class="evenTableCell"><A href="http://www.handbook.unsw.edu.au/un
....
```

You can run some tests on your script like this:

```
$ ~cs2041/bin/autotest lab03 courses.sh
```

Also do your own testing!

Finalising

You must show your solutions to your tutor and be able to explain how they work. Once your tutor has discussed your answers with you, you should submit them using:

```
$ give cs2041 lab03 digits.sh echon.sh file_sizes.sh [courses.sh]
```

Only submit `courses.sh` if you attempt the challenge exercise.

Whether you discuss your solutions with your tutor this week or next week, you must submit them before the above deadline.