Week 05 Weekly Test Questions

Test Conditions

These questions must be completed under self-administered exam-like conditions. You must time the test yourself and ensure you comply with the conditions below.

- You may complete this test in CSE labs or elsewhere using your own machine.
- You may complete this test at any time before **Week 7 Thursday 21:00:00**.
- Weekly tests are designed to act like a past paper - to give you an idea of how well you are progressing in the course, and what you need to work on. Many of the questions in weekly tests are from past final exams.
- Once the first hour has finished, you must submit all questions you've worked on.
- You should then take note of how far you got, which parts you didn't understand.
- You may choose then to keep working and submit test question anytime up to Week 7 Thursday 21:00:00
- However the maximum mark for any question you submit after the first hour will be 50%

You may access this **language documentation** while attempting this test:

- manual entries, via the *man* command.
- Texinfo pages, via the *info* command.
- Bash documentation, via the `help` build-in.
- Python documentation, via the `python3 -c 'help()'` command.
- Shell/Regex quick reference
- Python quick reference
- full Python 3.9 documentation

Any violation of the test conditions will results in a mark of zero for the entire weekly test component.

Getting Started

Set up for the test by creating a new directory called `test05` and changing to this directory.

```
$ mkdir test05
$ cd test05
```

There are some provided files for this test which you can fetch with this command:

```
$ 2041 fetch test05
```

If you're not working at CSE, you can download the provided files as a zip file or a tar file.

Test Complete!

Your time for this test has finished. You must submit your work now. You should reflect on how you went in this hour, and discuss with your tutor if you have concerns. You may choose to keep working, but the maximum mark for any questions you submit later will be 50%.

WEEKLY TEST QUESTION:

Hello Files

Write a POSIX-compatible shell script, `hello_files.sh` which takes 2 arguments.

The first argument will be positive integer, ***n***.

The second argument will be a string, ***name***.

Your program should create ***n*** files.

The names of these files should be **`hello1.txt`** .. **`hello`*`n`*`.txt`** .

Each file should have the same contents, a single line: **`hello`** ***name***

For example:

```
$ ls hello*.txt
ls: cannot access 'hello*.txt': No such file or directory
$ ./hello_files.sh 3 Andrew
$ ls hello*.txt
hello1.txt  hello2.txt  hello3.txt
$ wc hello*.txt
 1  2 13 hello1.txt
 1  2 13 hello2.txt
 1  2 13 hello3.txt
 3  6 39 total
$ cat hello1.txt
hello Andrew
$ cat  hello2.txt
hello Andrew
$ cat  hello3.txt
hello Andrew
$ ./hello_files.sh 100 Brittany
$ ls  hello*.txt|wc -l
100
$ cat hello100.txt
hello Brittany
$ cat hello42.txt
hello Brittany
$ cat hello1.txt
hello Brittany
```

HINT:

Make the first line of your shell-script `#!/bin/dash`

NOTE:

You are not permitted to use external programs such as `grep` , `sort` , `uniq` , ...
In particular you are not permitted to use the external program: **seq**.

You are permitted to use built-in shell arithmetic and other built-in shell features including:

`cd`	`if`	`test`
`echo`	`pwd`	`while`
`exit`	`read`	`[`
`for`	`shift`	

Note most of the above built-in shell features features are not useful for this problem.

You may not use non-POSIX-compatible shell features such as bash extensions.
Your script must work when run by `/bin/dash` on a CSE system.
You are not permitted to rely on the extra features provided by `/bin/bash` or `/bin/sh` .
You can assume anything that works with the version of `/bin/dash` on CSE systems is POSIX compatible.

You may not use Perl, C, Python, or any other language.

You can assume the files do not exist already.

No error checking is necessary.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 2041 autotest hello_files
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 test05_hello_files hello_files.sh
```

WEEKLY TEST QUESTION:

Print the file with most lines

We what to know which of a set of files has the most lines.

Write a POSIX-compatible shell script `most_lines.sh` which given one of more filenames as argument, prints which file has the most lines.

For example

```
$ seq 1 5 >five_lines.txt
$ cat five_lines.txt
1
2
3
4
5
$ seq 1 10 >ten_lines.txt
$ seq 1 100 >hundred_lines.txt
$ ./most_lines.sh ten_lines.txt hundred_lines.txt five_lines.txt
hundred_lines.txt
```

NOTE:

Your program can assume it is given 1 or more valid filenames as arguments.

Your program should print one line of output.
This line should contain only one of the filename is was given

If multiple files have the most lines your program may print any of them.

Your program can assume files contain only ASCII

Your program can every assume every line in these files is terminated by a single '\n' character

You are permitted to use external programs such as `wc` .

Make the first line of your shell-script `#!/bin/dash`

You are permitted to use built-in shell features.

You may not use non-POSIX-compatible shell features such as bash extensions.
Your script must work when run by `/bin/dash` on a CSE system.
You are not permitted to rely on the extra features provided by `/bin/bash` or `/bin/sh` .
You can assume anything that works with the version of `/bin/dash` on CSE systems is POSIX compatible.

You may not use Perl, C, Python, or any other language.

No error checking is necessary.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 2041 autotest most_lines
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 test05_most_lines most_lines.sh
```

WEEKLY TEST QUESTION:

List Identical Files in Shell

Write a POSIX-compatible shell script, **ls_identical.sh** which takes the pathnames of 2 directories as argument.

It should print in alphabetical order the names of all files which occur in both directories, and have exactly the same contents in both directories.

Files must have the same name in both directories and the same contents for their name to be printed.

Do not print the names of files with same contents but different names in both directories.

For example:

```
$ unzip directory.zip
Archive:  directory.zip
   creating: directory1/
   creating: directory2/
```

```
$ ls_identical.sh directory1 directory2
empty.txt
same.txt
```

> NOTE:
>
> Your program can assume it is given 2 valid directory names as arguments.
>
> Your program can assume file names do not start with '.'.
>
> Your program can assume files contain only ASCII
>
> You are permitted to use external programs such as `diff` .
>
> Make the first line of your shell-script `#!/bin/dash`
>
> You are permitted to use built-in shell features.
>
> You may not use non-POSIX-compatible shell features such as bash extensions.
> Your script must work when run by `/bin/dash` on a CSE system.
> You are not permitted to rely on the extra features provided by `/bin/bash` or `/bin/sh` .
> You can assume anything that works with the version of `/bin/dash` on CSE systems is POSIX compatible.
>
> You may not use Perl, C, Python, or any other language.
>
> No error checking is necessary.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 2041 autotest shell_ls_identical
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 test05_shell_ls_identical ls_identical.sh
```

Submission

When you are finished each exercise make sure you submit your work by running **give**.

You can run **give** multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Week 7 Thursday 21:00:00** to complete this test.

Automarking will be run by the lecturer several days after the submission deadline for the test, using test cases that you haven't seen: different to the test cases `autotest` runs for you.

Hint: do your own testing as well as running `autotest`

Test Marks

After automarking is run by the lecturer you can view it here the resulting mark will also be available via via give's web interface or by running this command on a CSE machine:

```
$ 2041 classrun -sturec
```

The test exercises for each week are worth in total 1 marks.

Each test is worth 1.7 marks, and will be automarked. Your total mark for the tests component is computed as a sum of your best 6 of 8 test marks.

For all enquiries, please email the class account at cs2041@cse.unsw.edu.au