Week 04 Weekly Test Sample Answers

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 5 Thursday 18:00:00**.
- 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 5 Thursday 18:00:00
- 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 `test04` and changing to this directory.

```
$ mkdir test04
$ cd test04
```

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

```
$ 2041 fetch test04
```

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

WEEKLY TEST QUESTION:

Create A File of Integers In Shell

Write a POSIX-compatible shell script, `create_integers_file.sh` which takes 3 arguments.

The first & second arguments will specify a range of integers.

The third argument will specify a filename.

Your program should create a file of this name containing the specified integers.

For example:

```
$ ./create_integers_file.sh 40 42 fortytwo.txt
$ cat fortytwo.txt
40
41
42
$ ./create_integers_file.sh 1 5 a.txt
$ cat a.txt
1
2
3
4
5
$ ./create_integers_file.sh 1 1000 1000.txt
$ wc 1000.txt
1000 1000 3893 1000.txt
```

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.

No error checking is necessary.

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

```
$ 2041 autotest shell_create_integers_file
```

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

```
$ give cs2041 test04_shell_create_integers_file create_integers_file.sh
```

SOLUTION:

Sample solution for `create_integers_file.sh`

```
#!/bin/dash

start="$1"
finish="$2"
filename="$3"

i="$start"
while [ "$i" -le "$finish" ]; do
    echo "$i"
    i=$((i + 1))
done > "$filename"
```

WEEKLY TEST QUESTION:

Change the Suffix of HTML Files

Our new web server requires all HTML files have the suffix `.html` . Unfortunately we have many HTML files named with the suffix `.htm` .

Write a POSIX-compatible shell script `htm2html.sh` which changes the name of all files with the suffix `.htm` in the current directory to have the suffix `.html` . For example:

```
$ touch index.htm small.htm large.htm
$ ls *.htm*
index.htm  large.htm  small.htm
$ ./htm2html.sh
$ ls *.htm*
index.html  large.html  small.html
```

Your script should stop with EXACTLY the error message shown below and exit status 1 if the `.html` file already exists. For example:

```
$ touch andrew.htm andrew.html
$ ./htm2html.sh
andrew.html exists
```

NOTE:

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

YMake 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 other than described above is necessary.

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

```
$ 2041 autotest htm2html
```

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

```
$ give cs2041 test04_htm2html htm2html.sh
```

SOLUTION:

Sample solution for `htm2html.sh`

```
#!/bin/dash

for file in *.htm; do
    new_file_name="$file"l
    if [ -e "$new_file_name" ]; then
        echo "$new_file_name exists" 1>&2
        exit 1
    else
        mv "$file" "$new_file_name"
    fi
done

exit 0
```

WEEKLY TEST QUESTION:

Checking for Missing Include Files

We need check a large number of C programs for missing include files.

Write a POSIX-comaptible shell script `missing_include.sh` which is give one of more filenames as argument. The files will contain C code.

`missing_include.sh` should print a message if any file included by the C program is not present in the current directory.

Reminder C include lines are of this form:

```
#include "file.h"
```

For example:

```
$ cat a.c
#include <stdio.h>

#include "a.h"
#include "b.h"
#include "input.h"

int a(void){
    return 42;
}
$ cat b.c
#include <stdio.h>

#include "b.h"
#include "c.h"
#include "d.h"
#include <string.h>
#include "global.h"

int b(void){
    return b.c;
}
$ ./missing_include.sh a.c
a.h included into a.c does not exist
input.h included into a.c does not exist
$ ./missing_include.sh b.c
c.h included into b.c does not exist
global.h included into b.c does not exist
$ ./missing_include.sh a.c b.c
a.h included into a.c does not exist
input.h included into a.c does not exist
c.h included into b.c does not exist
global.h included into b.c does not exist
```

Do not check includes with angle brackets (<>). For example do not check this line:

```
#include <stdio.h>
```

NOTE:

You can assume filenames do not contain whitespace or glob characters (`*[]?`).

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

YMake 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 missing_include
```

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

```
$ give cs2041 test04_missing_include missing_include.sh
```

SOLUTION:

Sample solution for `missing_include.sh`

```
#!/bin/dash

for c_file in "$@"; do
    include_files=$(
        grep -E '^#include\s*"' "$c_file" |
        sed '
            s/[^"]*"//
            s/"*$//
        '
    )

    # depends on include filesname not containing white-space or glob characters
    for include_file in $include_files; do
        if [ ! -r "$include_file" ]; then
            echo "$include_file included into $c_file does not exist"
        fi
    done
done
```

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 5 Thursday 18:00:00** 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.

The best 6 of your 8 test marks for weeks 3-10 will be summed to give you a mark out of 9.