Week 03 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 **Saturday 27 June 21:00**.
- 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 Saturday 27 June 21:00
- 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:

- Shell/Regex/Perl quick reference
- full Perl documentation

You may also access manual entries (the `man` command).

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

Set up for the test by creating a new directory called `test03`, changing to this directory, and fetching the provided code by running these commands:

```
$ mkdir test03
$ cd test03
$ 2041 fetch test03
```

Or, if you're not working on CSE, you can download the provided code 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:

Finding the Most Common First Name

We need to know the most common first name for each year's COMP[29]041 students.

We have enrollment data in this format:

```
$ wget https://cgi.cse.unsw.edu.au/~cs2041/20T2/activities/first_name/files.ln/enrollments.txt
$ head enrollments.txt
COMP1511|5013566|Xin, Mackenzie Darren                          |3648/2|COMPI1 MTRNAH|071.800|18s2|19910428|M
COMP9902|5079970|Park, Xue Hannah Vanessa                       |8543  |ELECAH       |079.333|18s2|19900209|F
COMP1511|5059072|Chung, Michael Jia Tianyu                      |3778/1|COMPCS       |057.250|18s2|19990801|M
COMP1521|5060774|Lim, Stephanie Lauren                          |3785/1|COMPA1       |000.000|18s2|19890113|F
COMP1531|5060774|Lim, Stephanie Lauren                          |3785/1|COMPA1       |000.000|18s2|19890113|F
COMP2521|5060774|Lim, Stephanie Lauren                          |3785/1|COMPA1       |000.000|18s2|19890113|F
COMP9020|5060538|Bi, Samuel Shiyu                               |6021  |COMPA1       |078.125|18s2|19911004|M
COMP9021|5060538|Bi, Samuel Shiyu                               |6021  |COMPA1       |078.125|18s2|19911004|M
COMP9902|5072116|Hu, Kai Zhi Patrick                            |3707/1|SENGAH       |070.750|18s2|19930424|M
COMP1511|5036926|Fang, Rebecca Lauren                           |8543  |COMPCS       |000.000|18s2|20000921|F
```

Write a shell script `first_name.sh` which takes the name of a file as a command-line argument, The file will contain enrolment data in the above format. `first_name.sh` should print a single line of output. This line should contain only the most common first name for COMP[29]041 students in the enrollment data. For example:

```
$ ./first_name.sh enrollments.txt
Vanessa
```

You can assume there will not be multiple first names which are equally common.

You can assume the data will be correctly formatted

No error checking is necessary.

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

```
$ 2041 autotest first_name
```

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

```
$ give cs2041 test03_first_name first_name.c
```

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 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
```

You can assume the current directory contains at last one `.htm` file

No error checking other then 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 test03_htm2html htm2html.sh
```

WEEKLY TEST QUESTION:

Checking for Missing Include Files

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

Write a 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:

```
$ wget https://cgi.cse.unsw.edu.au/~cs2041/20T2/activities/missing_include/file.cp/c_files.zip
$ unzip c_files.zip
Archive:  c_files.zip
  inflating: a.c
  inflating: b.c
 extracting: b.h
 extracting: d.h
$ ls *.[ch]
a.c  a.h  b.c  c.h
$ 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>

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

You can assume filenames do not contain spaces.

You do not have to check files for the C library with angle brackets (<>). For example you do not have to check this line:

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

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 test03_missing_include missing_include.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 **Saturday 27 June 21:00** 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.