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 **Thursday 11 October 23:59:59**
- The maximum time allowed for this test is 1 hour + 5 minutes reading time.
- You may first use 5 minutes to read the questions (no typing)
- You then must complete the test within 1 hour and submit your answers with give.
- You must complete the questions alone - you can not get help in any way from any person.
- You can not access your previous answers to lab or tut questions.
- You can not access web pages or use the internet in any way.
- You can not access books, notes or other written or online materials.
- You can not access your own files, programs, code ...
- You can not access COMP2041 course materials except for language documentation linked below.

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

- Shell/Regex/Perl quick reference
- Javascript quick reference
- full Perl documentation
- Python quick reference
- C quick reference
- full Python 3.6 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.

Print Your Median Argument

Write a Perl program **median_number.pl** that given positive integers as command line arguments prints the median (middle) value.
Your program can assume is given an odd number of arguments.

Your program can assume all its arguments are positive integers.

For example:

```
$ ./median_number.pl 1 333 42
42
$ ./median_number.pl 3 4 2 1 7 6 5
4
$ ./median_number.pl 15 15 8 11 8
11
$ ./median_number.pl 42 42 244 244 42
42
```

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

```
$ 2041 autotest median_number
```

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

```
$ give cs2041 test10_median_number median_number.pl
```

Sample solution for `median_number.pl`

```
#!/usr/bin/perl -w

# print the median value of positive numbers supplied as arguments
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

die "$0: number of numbers supplied as arguments must be odd\n" if @ARGV % 2 != 1;

my @sorted_numbers = sort {$b <=> $a} @ARGV;
my $median_index = @sorted_numbers / 2;
my $median_number = $sorted_numbers[$median_index];
print "$median_number\n";
```

Alternative solution for `median_number.pl`

```
#!/usr/bin/perl -w

# print the median value of positive numbers supplied as arguments
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution
# one line version

print(((sort {$b <=> $a} @ARGV)[@ARGV/2]),"\n");
```

List Identical Files in Shell

Write a Shell program, **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.

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:

```
$ ls_identical.1.sh directory1 directory2
```

You can assume file names do not start with '.'.
Your answer must be Shell. You can not use other languages such as Perl, Python or C.

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 test10_shell_ls_identical ls_identical.sh
```

Sample solution for `ls_identical.sh`

```
#!/bin/bash

# list identical files in the two directories given as arguments
# files starting with . are ignored
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

if test "$#" != 2
then
    echo "Usage: $0 <directory1> <directory2>" 1>&2
    exit 1
fi

directory1="$1"
directory2="$2"

for file in "$directory1"/*
do
    # compare file to coressponding file in directory2
    # any error message e.g. if file not present in directory2
    # is directed to /dev/null

    if diff "$file" "$directory2" >/dev/null 2>&1
    then
        basename "$file"
    fi
done
```

Alternative solution for `ls_identical.sh`

```
#!/bin/bash

# list identical files in the two directories given as arguments
# files starting with . are ignored
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution
# terse less readable version

#test "$#" != 2 &&  echo "Usage: $0 <directory1> <directory2>" 1>&2 && exit 1
for file in "$1"/*
do
    diff "$file" "$2" >/dev/null 2>&1 && basename "$file"
done
```

List Identical Files in Perl

Write a Perl program, **ls_identical.pl** 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.

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:

```
$ mkdir directory1 directory2
$ echo hello >directory1/same.txt
$ echo hello >directory2/same.txt
$ echo hello >directory1/different.txt
$ echo world >directory2/different.txt
$ echo hello >directory1/one.txt
$ echo hello >directory2/two.txt
$ touch directory1/empty.txt directory2/empty.txt
$ ls directory1
different.txt
empty.txt
one.txt
same.txt
$ ls directory2
different.txt
empty.txt
same.txt
two.txt
$ ls_identical.pl directory1 directory2
empty.txt
same.txt
```

You can assume file names do not start with '.'.
Your answer must be Perl only. You can not use other languages such as Shell, Python or C.

You may not run external programs, e.g. via system or backquotes. for example, you can't run `diff`.

You may use any Perl module installed on CSE systems.

No error checking is necessary.

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

```
$ 2041 autotest perl_ls_identical
```

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

```
$ give cs2041 test10_perl_ls_identical ls_identical.pl
```

Sample solution for `ls_identical.pl`

```
#!/usr/bin/perl -w

# list identical files in the two directories given as arguments
# files starting with . are ignored
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

sub main() {
    die "Usage: $0 <directory1> <directory2>\n" if @ARGV != 2;
    for $file1 (sort glob "$ARGV[0]/*") {
        my $basename = $file1;
        $basename =~ s/.*\///;
        my $file2 = "$ARGV[1]/$basename";
        if (identical_files($file1, $file2)) {
            print "$basename\n" if -r $file2;
        }
    }
}

sub identical_files {
    my ($file1, $file2) = @_;
    return 0 if !-r $file1 or !-r $file2;
    return read_file($file1) eq read_file($file2);
}

sub read_file {
    my ($file) = @_;
    open my $f, '<', $file or die "$0: can not open file $file: $!\n";
    my @lines = <$f>;
    close $f;
    return join "", @lines;
}

main()
```

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 **Thursday 11 October 23:59:59** 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`)

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