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 **Wednesday 26 September 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.

Unique Echo

Write a Perl program **uniq_echo.pl** that prints its command-line argument to standard output, similar to **echo** command in Shell, except only the first occurrence of any argument should be printed, Repeat occurrences should not be printed.

```
$ ./uniq_echo.pl echo echo echo
echo
$ ./uniq_echo.pl bird cow bird cow fish bird cow fish bird
bird cow fish
$ ./uniq_echo.pl how much wood would a woodchuck chuck
how much wood would a woodchuck chuck
$ ./uniq_echo.pl d c b d c a a d
d c b a
$ ./uniq_echo.pl
```

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.

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

```
$ 2041 autotest uniq_echo
```

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

```
$ give cs2041 test08_uniq_echo uniq_echo.pl
```

Sample solution for `uniq_echo.pl`

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

# print command line arguments unless they are a repeat
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

foreach $argument (@ARGV) {
    next if $arguments_seen{$argument};
    print " " if %arguments_seen;
    print $argument;
    $arguments_seen{$argument} = 1;
}

print "\n";
```

Alternative solution for `uniq_echo.pl`

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

# print command line arguments unless they are a repeat
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution
# more concise less readable version

my %seen; # avoid warning
print join(" ", grep(!$seen{$_}++, @ARGV)), "\n"
```

Snap N

Write a Perl program which reads lines from its input until it reads an line that has been entered *n* times.
Your program should then print "Snap: " followed by the line.

Your program will be given *n* as a command line argument. Your program should print nothing if the end-of-input is reached before any line is repeated *n* times.

You can assume the lines are ASCII, you should not assume anything else about the lines.

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:

```
$ ./snap_n.pl 2
hi
how
are
you
hi
Snap: hi
```

```
$ ./snap_n.pl 2
hi
hi hi
hi hi hi
hi hi hi hi
hi hi
Snap: hi hi
```

```
$ ./snap_n.pl 4
Hello World
Line 2
Hello World
Line 3
Hello World
Line 4
Hello World
Snap: Hello World
```

No error checking is necessary.

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

```
$ 2041 autotest snap_n
```

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

```
$ give cs2041 test08_snap_n snap_n.pl
```

Sample solution for `snap_n.pl`

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

# Stop after a line oN STDIN is seen n times, n is supplied as a command line argument
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

die "Usage: $0 <n>" if @ARGV != 1;

my $snap_after_n_repeats = $ARGV[0];
my %line_repeats;

while ($line = <STDIN>) {
    $line_repeats{$line}++;

    if ($line_repeats{$line} >= $snap_after_n_repeats) {
        print "Snap: $line";
        last;
    }
}
```

Print the lines of A File Sorted on Length

Write a Perl program, `sort_file_lines.pl` which is given one argument, a file name.
Your program should print the lines of the file in order of length, shortest to longest.

Lines of equal length should be sorted alphabetically.

You can asusme the lines in the file contain ASCII. Yoy should not assume anything else about the lines in the file.

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.

No error checking is necessary.

For example:

```
$ cat file.txt
tiny
short line
medium line
longggggggg line
a equal line
b equal line
c equal line
even longgggggggggggggggggggggggggggggggggggggggggggerr
$ sort_file_lines.pl file.txt
tiny
short line
medium line
a equal line
b equal line
c equal line
longggggggg line
even longgggggggggggggggggggggggggggggggggggggggggggerr
```

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

```
$ 2041 autotest sort_file_lines
```

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

```
$ give cs2041 test08_sort_file_lines sort_file_lines.pl
```

Sample solution for `sort_file_lines.pl`

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

# print the lines of a files sorted on length, shortest to longest
# if lines are of equal length they sorted alphabetically
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

sub compare {
    my $a_length = length $a;
    my $b_length = length $b;
    if ($a_length == $b_length) {
        return $a cmp $b;
    } else {
        return $a_length <=> $b_length;
    }
}

die "Usage $0: <file>\n" if @ARGV != 1;
open my $f, "<", $ARGV[0] or die "$0: can not open $ARGV[0]: $!\n";
@lines = <$f>;
close $f;

@sorted_lines = sort compare @lines;

print @sorted_lines;

exit 0;
```

Alternative solution for `sort_file_lines.pl`

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

# print the lines of a files sorted on length, shortest to longest
# if lines are of equal length they sorted alphabetically
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution
# more concise but less readable solution

print sort {length $a <=> length $b || $a cmp $b} <>;
```

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 **Wednesday 26 September 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