Week 06 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 **Tuesday 21 July 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 Tuesday 21 July 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 `test06`, changing to this directory, and fetching the provided code by running these commands:

```
$ mkdir test06
$ cd test06
$ 2041 fetch test06
```

Or, if you're not working on CSE, you can download the provided code as a zip file or a tar file.

WEEKLY TEST QUESTION:

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 test06_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"
```

WEEKLY TEST QUESTION:

Snap N

Write a Perl program which reads lines from its input until it reads an line that has been entered Ṏ times.

Your program should then print "Snap: " followed by the line.

Your program will be given Ṏ as a command line argument. Your program should print nothing if the end-of-input is reached before any line is repeated Ṏ 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 test06_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;
    }
}
```

WEEKLY TEST QUESTION:

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

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 test06_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 **Tuesday 21 July 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.

COMP(2041|9044) 20T2: Software Construction is brought to you by
the School of Computer Science and Engineering
at the University of New South Wales, Sydney.
For all enquiries, please email the class account at cs2041@cse.unsw.edu.au
CRICOS Provider 00098G