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 25 July 21:59:59**
- 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.

Echo If Consecutive Three Vowels

Write a Perl program **three_vowel_echo.pl** that prints its command-line argument to standard output, similar to **echo** command in Shell, except arguments should be printed only if they contain 3 consecutive vowels.
Your program can assume vowels are there are 5 vowel {'a', 'e', 'i', 'o', 'u'} and their upper-case equivalents {'A', 'E', 'I', 'O', 'U'}

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.

```
$ ./three_vowel_echo.pl Pompeii Rome Aeolian Florence
Pompeii Aeolian
$ ./three_vowel_echo.pl

$ ./three_vowel_echo.pl an anxious bedouin beauty booed an ancient zoologist
anxious bedouin beauty booed
$ ./three_vowel_echo.pl abstemiously adenocarcinomatous Hawaiian Eoanthropus
abstemiously Hawaiian Eoanthropus
```

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

```
$ 2041 autotest three_vowel_echo
```

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

```
$ give cs2041 test07_three_vowel_echo three_vowel_echo.pl
```

Sample solution for `three_vowel_echo.pl`

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

# print command line arguments which contain 3 consecutive vowels
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

my $n_printed = 0;
foreach $argument (@ARGV) {
    if ($argument =~ /[aeiou][aeiou][aeiou]/i) {
        print " " if $n_printed++;
        print $argument;
    }
}
print "\n";
```

Alternative solution for `three_vowel_echo.pl`

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

# print command line arguments which contain 3 consecutive vowels
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution
# more conise less readable version

print join(" ", grep(/[aeiou][aeiou][aeiou]/i, @ARGV)), "\n"
```

Print the Middle Lines of A File

Write a Perl program middle_lines.pl which prints the middle line(s) in a file.
If the file contains an odd number of lines it should print one line.

If the file contains an even number of lines it should print two lines.

If the file contains no lines it should print nothing.

You can assume one and only one file is given as argument and that it exists and it is readable.

For example:

```
$ cat odd.txt
line 0
line 1
line 2
line 3
line 4
$ ./middle_lines.pl odd.txt
line 2
$ cat even.txt
line 0
line 1
line 2
line 3
line 4
line 5
$ ./middle_lines.pl even.txt
line 2
line 3
$ ./middle_lines.pl /dev/null
```

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 middle_lines
```

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

```
$ give cs2041 test07_middle_lines middle_lines.pl
```

Sample solution for **middle_lines.pl**

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

# print middle lines of a file
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

open F, '<', $ARGV[0] or die;
my @lines = <F>;
close F;

if (!@lines) {
    # no lines
} elsif (@lines % 2 == 1) {
    # odd number of lines
    print $lines[$#lines/2];
} else {
    # even number of lines
    print $lines[$#lines/2];
    print $lines[$#lines/2 + 1];
}
```

Alternative solution for **middle_lines.pl**

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

# print middle lines of a file
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution
# more conise less readable version

my @lines = <>;
print @lines[(@lines-1)/2..((@lines-1)/2 + 1 - (@lines%2))] if @lines;
```

Print the Line(s) from Stdin With the Largest Number

Write a Perl program **largest_numbered_line.pl** that read lines from standardinput until end-of-input. It should then print the line(s) which contained the largest number.
You can assume numbers do not contain white space, commas or other extra characters.

You can assume numbers are only in decimal format.

You can assume numbers are not in scientific/exponential format.

Lines may contain multiple numbers and they may contain any number of any character between the numbers.

If the largest number occurs on multiple lines you should print all of the lines in the order they occurred.

If no line contains a number, your program should print nothing.

```
$ ./largest_numbered_line.pl
I spent $ 15.50 for
3.3 kg apples yesterday.
2000 is a leap year.
Ctrl-D
2000 is a leap year.
```

```
$ ./largest_numbered_line.pl
two2 four4 eight8 sixteen16
1 sixteen-and-half 16.5  1
11 12 13
Ctrl-D
1 sixteen-and-half 16.5  1
```

```
$ ./largest_numbered_line.pl
the quick brown f42ox
4 9 42 2 4
1 2 3 4 42.0
no forty two
last 42
Ctrl-D
the quick brown f42ox
4 9 42 2 4
1 2 3 4 42.0
last 42
```

```
$ ./largest_numbered_line.pl
a 0.01
b .5
c -0.9
Ctrl-D
b .5
```

```
$ ./largest_numbered_line.pl
a -.5
b -5
c --90--
Ctrl-D
a -.5
```

```
$ ./largest_numbered_line.pl
I love programming in Perl
but I like Python better.
Ctrl-D
```

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 largest_numbered_line
```

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

```
$ give cs2041 test07_largest_numbered_line largest_numbered_line.pl
```

Sample solution for **largest_numbered_line.pl**

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

# print line of a file containing largest number
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution

while ($line = <STDIN>) {
    my @line_numbers = $line =~ /\-?(?:\d+\.?\d*|\.\d+)/g;
    if (@line_numbers) {
        my $largest_line_number = (sort {$b <=> $a} @line_numbers)[0];
        push @numbers, $largest_line_number;
        push @lines, $line;
    }
}

if (@numbers) {
    my $largest_number = (sort {$b <=> $a} @numbers)[0];
    foreach $i (0..$#numbers) {
        if ($numbers[$i] == $largest_number) {
            print $lines[$i];
        }
    }
}
```

Alternative solution for `largest_numbered_line.pl`

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

# print line of a file containing largest number
# written by andrewt@unsw.edu.au as COMP[29]041 sample solution
# more concise less readable version

push @numbers, /\-?(?:\d+\.?\d*|\.\d+)/g foreach @lines = <STDIN>;
exit if !@numbers;
$largest = (sort {$b <=> $a} @numbers)[0];
(sort {$b <=> $a} /\-?(?:\d+\.?\d*|\.\d+)/g)[0] == $largest && print foreach @lines;
```

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 25 July 21:59:59** 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

The test exercises for each week are worth in total 1 mark.
The best 6 of your 8 test marks for weeks 3-10 will be summed to give you a mark out of 6.