Week 10 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 **Thursday 13 August 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 Thursday 13 August 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 `test10`, changing to this directory, and fetching the provided code by running these commands:

```
$ mkdir test10
$ cd test10
$ 2041 fetch test10
```

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:

Don't Repeat Yourself

Write a program **remove_repeats.pl** which takes 0 or more arguments and prints some of those arguments.

The first occurrence and only the first occurrence of any argument should be printed.

The arguments should be printed on a single line. A single space should separate each argument.

Your program can **NOT** assume the arguments with be in any particular order.

Make your program behave **exactly** as indicated by the examples below.

It must produce **exactly** the same output as below, except you may print an extra space at the end of the line if you wish.

Your program must be Perl.

For example:

```
$ remove_repeats.pl

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

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

```
$ 2041 autotest remove_repeats
```

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

```
$ give cs2041 test10_remove_repeats remove_repeats.pl
```

Sample solution for `remove_repeats.pl`

```perl
#!/usr/bin/perl -w
foreach $arg (@ARGV) {
    next if $seen{$arg};
    print "$arg ";
    $seen{$arg} = 1;
}
print "\n";
```

Alternative solution for `remove_repeats.pl`

```perl
#!/usr/bin/perl
my %seen;
print join(" ",grep(!$seen{$_}++, @ARGV)), "\n";
```

WEEKLY TEST QUESTION:

Well-Rounded Text

Write a program **text_round.pl**that copies its standard input to standard output but maps all numbers to their nearest whole number equivalent. For example, **0.667** would be mapped to **1**, **99.5** would be mapped to **100**, **16.35** would be mapped to **16**, and so on. All other text in the input should be transferred to the output unchanged.

A *number* is defined as a string containing some digit characters with an optional decimal point ('**.**') followed by zero or more additional digit characters.

For example **0**, **100**, **3.14159**, **1000.0**, **0.999** and **12345.** are all valid numbers.

For example, given this input:

```
I spent $15.50 for 3.3kg of apples yesterday.
Pi is approximately 3.141592653589793
2000 is a leap year, 2001 is not.
```

your program should produce this output:

```
I spent $16 for 3kg of apples yesterday.
Pi is approximately 3
2000 is a leap year, 2001 is not.
```

For example:

```
$ cat text_round_input.txt
I spent $15.50 for 3.3kg of apples yesterday.
Pi is approximately 3.141592653589793
2000 is a leap year, 2001 is not.
$ text_round.pl <text_round_input.txt
I spent $16 for 3kg of apples yesterday.
Pi is approximately 3
2000 is a leap year, 2001 is not.
```

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

```
$ 2041 autotest text_round
```

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

```
$ give cs2041 test10_text_round text_round.pl
```

Sample solution for `text_round.pl`

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

while ($line = <>) {
    my @numbers = $line =~ /(\d+\.\d+)/g;
    foreach $number (@numbers) {
        my $rounded_number = int($number + 0.5);
        $line =~ s/\b$number\b/$rounded_number/;
    }
    print $line;
}
```

Alternative solution for `text_round.pl`

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

while ($line = <>) {
    while ($line =~ /(\d+\.\d+)/) {
        my $number = $1;
        $number =~ /^(\d+)/;
        my $rounded_number = $1;
        if ($number =~ /\.[5-9]/) {
            $rounded_number++;
        }
        $line =~ s/$number/$rounded_number/;
    }
    print $line;
}
```

Alternative solution for `text_round.pl`

```
#!/usr/bin/perl -wp
s/(\d+\.\d+)/int($&+0.5)/eg;
```

WEEKLY TEST QUESTION:
Replace Those References

Write a program **reference.pl** that reads lines of text from its standard input and prints them to its standard output. Except for lines which contain with a '#' character followed by a positive integer.

Lines of the form **#n** (where **n** is an integer value), should be replaced this by the **n**'th line of input.

This transformation only applies to lines which start with a # character, followed by the digits of a positive integer and then the newline character. No other characters appear on such lines.

You may assume:

Lines are numbered starting from 1,

There are no more than 100 lines in the input,

No line is more than 80 characters long,

All **n** values are valid input line numbers,

No **n** values refer to other #**n** lines.

For example:

```
$ cat reference_input.txt
line A
line B
line C
#7
line D
#2
line E
$ reference.pl <reference_input.txt
line A
line B
line C
line E
line D
line B
line E
```

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

```
$ 2041 autotest reference
```

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

```
$ give cs2041 test10_reference reference.pl
```

Sample solution for `reference.pl`

```
#!/usr/bin/perl -w
@a = <STDIN>;
foreach $i (0..$#a) {
    if ($a[$i] =~ /^#(\d+)/) {
        $a[$i] = $a[$1 - 1];
    }
}
print @a;
```

Alternative solution for `reference.pl`

```
#!/usr/bin/perl -w
@a = <STDIN>;
/^#(\d+)/ and $_ = $a[$1 - 1] foreach @a;
print @a;
```

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 13 August 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.