Week 06 Tutorial

Sample Answers

1. A citizen science project monitoring whale populations has files containing large numbers of whale observations. Each line in these files contains:

   - the date the observation was made
   - the number of whales in the pod ("pod" is the collective number for a group of whales)
   - the species of whale

   For example:

```
$ cat whale_observations.txt
01/09/18 21 Southern right whale
01/09/18  5 Southern right whale
02/09/18  7 Southern right whale
05/09/18  4 Common dolphin
05/09/18  9 Pygmy right whale
05/09/18  4 Striped dolphin
05/09/18 35 Striped dolphin
05/09/18  4 Blue whale
```

   Write a Perl program `./merge_whales.pl` which reads a file of whale observations and prints them to its standard output, merging adjacent counts from the same day of the same species. For example:

```
$ ./merge_whales.pl  whale_observations.txt
01/09/18 26 Southern right whale
02/09/18 7 Southern right whale
05/09/18 4 Common dolphin
05/09/18 9 Pygmy right whale
05/09/18 39 Striped dolphin
05/09/18 4 Blue whale
```

   Sample Perl solution

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

for $filename (@ARGV) {
    open my $f, $filename or die "Can not open $filename\n";

    my $current_date = "";
    my $current_count = 0;
    my $current_species = "";

    while ($line = <$f>) {
        if ($line =~ /^(\S+)\s+(\d+)\s+(.+)\s*$/) {
            my $date = $1;
            my $count = $2;
            my $species = $3;

            if ($species eq $current_species && $date eq $current_date) {
                $current_count += $count;
            } else {
                print "$current_date $current_count $current_species\n" if $current_count;
                $current_date = $date;
                $current_count = $count;
                $current_species = $species;
            }

        } else {
            print "Sorry couldn't parse: $line\n";
        }
    }

    print "$current_date $current_count $current_species\n" if $current_count;
}
```

2. Write a Perl program `./whale_last_seen.pl` which reads a file of whale observations in the same format as the last question and prints the species in alphabetical order, with the date they were last seen.

```
$ ./whale_last_seen.pl  whale_observations.txt
Blue whale 05/09/18
Common dolphin 05/09/18
Pygmy right whale 05/09/18
Southern right whale 02/09/18
Striped dolphin 05/09/18
```

You can assume the file is in chronological order, so the last line in the file for a species will be the last date for the species.

Sample Perl solution

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

for $filename (@ARGV) {
    open my $f, $filename or die "Can not open $filename\n";

    while ($line = <$f>) {
        if ($line =~ /^(\S+)\s+\d+\s+(.+)\s*$/) {
            my $date = $1;
            my $species = $2;

            $last_seen{$species} = $date;
        } else {
            print "Sorry couldn't parse: $line\n";
        }
    }

    foreach $species (sort keys %last_seen) {
        print "$species $last_seen{$species}\n";
    }
}
```

3. Write a Perl program, `phone_numbers.pl` which given the URL of a web page fetches it by running *wget* and prints any strings that might be phone numbers in the web page.
   Assume the digits of phone numbers may be separated by zero or more spaces or hyphens ('-') and can contain between 8 and 15 digits inclusive.

   You should print the phone numbers one per line with spaces & hyphens removed.

   For example

```
$ ./phone_numbers.pl http://www.unsw.edu.au
20151028
11187777
841430912571345
413200225
61293851000
57195873179
```

   Sample Perl solution

```
#!/usr/bin/perl -w
# there are perl libraries which provide a  better way to fetch web pages
foreach $url (@ARGV) {
    open my $f, '-|', "wget -q -O- $url" or die;
    while ($line = <$f>) {
        @numbers = split /[^\d\- ]/, $line;
        foreach $number (@numbers) {
            $number =~ s/\D//g;
            print "$number\n" if length $number >= 8 && length $number <= 15;
        }
    }
    close $f;
}
```

   Another sample Perl solution for phone_numbers.1.pl

```
#!/usr/bin/perl -w
# there are perl libraries which provide a  better way to fetch web pages
foreach $url (@ARGV) {
    open my $f, '-|', "wget -q -O- $url" or die;
    while ($line = <$f>) {
        @numbers = $line =~ /[\d\- ]+/g;
        foreach $number (@numbers) {
            $number =~ s/\D//g;
            print "$number\n" if length $number >= 8 && length $number <= 15;
        }
    }
    close $f;
}
```

   Python solution

```
#!/usr/bin/python
import sys, re, subprocess
# there are python libraries which provide a  better way to fetch web pages
for url in sys.argv[1:]:
    webpage = subprocess.Popen(["wget","-q","-O-",url], stdout=subprocess.PIPE).communicate()[0]
    for number in re.findall(r'[\d \-]+', webpage):
        number = re.sub(r'\D', '', number)
        if len(number) >= 8 and len(number) <= 15:
            print(number)
```

4. What does each of the following Perl code fragments print (no, don't just clip the lines and pass them to Perl, think about what they're doing):

a.
```
$x = 'x';
for ($i = 1; $i <= 3; $i++) {
    $x = "($x)";
}
print "$x\n";
```

It iterates three times through the loop, and each iteration wraps a pair of parentheses around what was there on the last iteration so it prints:

```
(((x)))
```

b.
```
@hi = split //,"hello";
for ($i = 0; $i < 4; $i++) {
    print $hi[$i];
}
print "\n";
```

The idiom `split //` splits a string into an array of individual characters (in this case `("h","e","l","l","o")`. The loop iterates over the first four of these characters. So it prints:

```
hell
```

c.
```
@vec = (10 .. 20);
print "@vec[1..3]\n";
```

The first statement produces an array containing the integers between 10 and 20 inclusive; the expression in the `print` statement takes a slice of this array from the 2nd to the 4th elements (remember that index values start at zero) so i prints:

```
11 12 13
```

d.
```
foreach $n (1..10) {
    last if ($n > 5);
    print "$n ";
    next if ($n % 2 == 0);
    print "$n ";
}
print "\n";
```

The loop iterates with `$n` set to the values from 1 to 10 inclusive. The `last` terminates the loop as soon as the value of `$n` exceeds five (cf. `break` in C or Java). The `next` starts the next iteration straight away whenever the test succeeds; which occurs for each even number, so that the evens only get printed once so it prints:

```
1 1 2 3 3 4 5 5
```

Revision questions

The remaining tutorial questions are primarily intended for revision - either this week or later in session.
Your tutor may still choose to cover some of the questions time permitting.

5. Write a Perl program `source_count.pl` which prints the number of lines of C source code in the current directory. In other words, this Perl program should behave similarly to `wc -l *.[ch]`. (Note: you are not allowed to use `wc` or other Unix programs from within the Perl script). For example:

```
$ ./source_count.pl
    383 cyclorana.c
    280 cyclorana.h
     15 enum.c
    194 frequency.c
    624 model.c
    293 parse.c
    115 random.c
     51 smooth.c
    132 util.c
     16 util.h
    410 waveform.c
   2513 total
```

Sample Perl solution

```
#!/usr/bin/perl -w
# written by andrewt@cse.unsw.edu.au for COMP2041
# count lines of C source code
$total = 0;
foreach $file (glob("*.[ch]")) {
    open my $f, '<', $file or die "Can not open $file: $!";
    @lines = <$f>;
    $n_lines = @lines;
    # Why wouldn't   $n_lines = <FILE>;   work?

    printf "%7d %s\n", $n_lines, $file;
    $total += $n_lines;
    close $f;
}
printf "%7d total\n", $total;
```

Sample Python solution

```
#!/usr/bin/python
# written by andrewt@cse.unsw.edu.au for COMP2041
# count lines of C source code
import glob
total = 0
for filename in glob.glob("*.[ch]"):
    with open(filename) as f:
        lines = f.readlines()
        n_lines = len(lines)
        print("%7d %s" % (n_lines, filename))
        total += n_lines
print("%7d total"%total)
```

6. Write a Perl program, `cut.pl` which takes three arguments, *n*, *m* and a file name. It should print characters *n*-*m* of each line of the file. For example:

```
$ ./cut.pl 1 8 file
```

should print the 8 characters of every line in `file`.

Implement a version of the program which invokes `/usr/bin/cut` and a version which performs the operations directly in Perl.

Sample Perl solution calling /usr/bin/cut

```
#!/usr/bin/perl -w
system "cut -c$ARGV[0]-$ARGV[1] $ARGV[2]";
```

Sample solution in Perl itself

```
#!/usr/bin/perl -w
die "Usage: $0 <n> <m> <file" if @ARGV != 3;
open my $f, '<', $ARGV[2] or die "$0: can not open $ARGV[2]: $!";
while ($line = <$f>) {
    chomp $line;
    @chars = split //, $line;
    print @chars[$ARGV[0]-1..$ARGV[1]-1], "\n";
}
```

Sample solution implementing more of cut in Perl

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

$delim = "\t";
if ($ARGV[0] =~ /-d./) {
    ($delim = $ARGV[0]) =~ s/-d//;
    shift;
}
if ($ARGV[0] =~ /-f*/) {
    ($flist = $ARGV[0]) =~ s/-f//;
    if ($flist eq "")
        { shift; $flist = $ARGV[0]; }
    @fields = split(/,/,$flist);
    shift;
}
while (<>) {
    chomp;
    @words = split /$delim/;
    @outs = ();
    $nf = $#words;
    foreach $f (@fields) {
        push @outs, $words[$f-1] if ($f <= $nf+1);
    }
    print join($delim,@outs)."\n";
}
```

Python solution calling /usr/bin/cut

```
#!/usr/bin/python
import subprocess, sys
print()
subprocess.call(["cut","-c%s-%s"%(sys.argv[1:3]), sys.argv[3]])
```

Sample solution in Python

```
#!/usr/bin/python

import sys

if len(sys.argv) != 4:
    sys.stderr.write("Usage: %s <n> <m> <file>\n" % sys.argv[0])
    sys.exit(1)

for line in open(sys.argv[3]):
    line = line.rstrip('\n')
    print(line[int(sys.argv[1])-1:int(sys.argv[2])])
```

7. Implement a Perl script to solve the marks-to-grades problem that was solved as a shell script in a previous tutorial. Reminder: the script reads a sequence of (studentID,mark) pairs from its standard input and writes (studentID,grade) pairs to its standard output. The input pairs are written on a single line, separated by spaces, and the output should use a similar format. The script should also check whether the second value on each line looks like a valid grade, and output an appropriate message if it doesn't. The script can ignore any extra data occuring after the mark on each line.
   Consider the following input and corresponding output to the program:

Input	**Output**

```
2212345 65
2198765 74
2199999 48
2234567 50 ok
2265432 99
2121212 hello
2222111 120
2524232 -1
```

```
2212345 CR
2198765 CR
2199999 FL
2234567 PS
2265432 HD
2121212 ?? (hello)
2222111 ?? (120)
2524232 ?? (-1)
```

Sample Perl solution

```
#!/usr/bin/perl -w
while (<>) {
    chomp;
    ($sid,$mark) = split;
    if ($mark !~ /^\d+$/) {
        $grade = "??";
    }
    else {
        $grade = "FL" if $mark >= 0 && $mark < 50;
        $grade = "PS" if $mark >= 50 && $mark < 65;
        $grade = "CR" if $mark >= 65 && $mark < 75;
        $grade = "DN" if $mark >= 75 && $mark < 85;
        $grade = "HD" if $mark >= 85 && $mark <= 100;
        $grade = "??" if $mark < 0 || $mark > 100;
    }
    $err = ($grade ne "??") ? "" : " ($mark)";
    print "$sid $grade$err\n";
}
```

8. Write a program `addressbook.pl` that reads two files `people.txt` and `phones.txt` containing data in CSV (comma-separated values) format and uses this data to print an address book in the format below:

```
$ cat people.txt
andrew,Andrew Taylor,42 Railway St,Petersham
arun,Arun Sharma,94 Leafy Close,Brisbane
gernot,Gernot Heiser,64 Trendy Tce,Newtown
jas,John Shepherd,16/256 Busy Rd,Alexandria
$ cat phones.txt
jas,home,9665 6432
arun,work,9385 5518
jas,work,9385 6494
arun,home,(07) 9314 6543
andrew,work,9385 4321
arun,mobile,0803 123 432
$ ./addressbook.pl
Andrew Taylor
42 Railway St, Petersham
Phones: 9385 4321 (work)

Arun Sharma
94 Leafy Close, Brisbane
Phones: 0803 123 432 (mobile), (07) 9314 6543 (home), 9385 5518
 (work)

Gernot Heiser
64 Trendy Tce, Newtown
Phones: ?
```

Assume that there are only three types of phone (mobile, home and work) and we always display them in that order.

Hint: because the phone types are fixed, login name and phone type together can be the key used to look up a number. In this situation some suitable separator is used to create a composite, unambiguous key for the hash.

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

# Build hash table of phone numbers, where keys
# are "person:phone-type" strings

open my $phones, '<', "phones.txt" or die "Can not open phones.txt:$!\n";

while ($line = <$phones>) {
    chomp $line;
    my ($id,$type,$number) = split /,/, $line;
    $phones{"$id:$type"} = $number;
}

close $phones;

# Iterate through People, displaying values
open my $people, '<', "people.txt" or die "Can not open people.txt:$!\n";

while ($line = <$people>) {
    chomp $line;
    my ($id,$name,$street,$suburb) = split /,/, $line;
    print "$name\n$street, $suburb\nPhones: ";
    my $nphones = 0;
    foreach $type ('mobile', 'home', 'work') {
        my $key = "$id:$type";
        if (defined($phones{$key})) {
            $num = $phones{$key};
            print ", " if ($nphones++ > 0);
            print "$num ($type)";
        }
    }
    print "?" if ($nphones == 0);
    print "\n\n";
}

close $people;
```