Examination questions for BINP13, 2013-10-31 (09.00 - 13.00).

Approximately 15p are required for passing the exam.

Part 1: Interprete Perl code

Question 1 (3p): Describe the variable used in the print statements below in terms of scalars, arrays, hashes and references to them.

```
2. print $a->{'Perl'};

1. print $a[1];

3. print $a;

4. print $a{'Ala'}

5. print $a->[2];

6. print ${$a};
```

Question 2 (1p): What is the output of the following program?

```
#! /usr/bin/perl -w
use strict;

my $mess1 = "I'm" . ' ' . "a";
my $mess2 = "Hobbit";
print $mess1, " $mess2", " fan!\n";
```

```
1 I'm a Hobbit fan!
```

Question 3 (2p): What is the output of the following program?

```
#! /usr/bin/perl -w
use strict;

my %tr = ('1' => ' ', '2' => ' it', '3' => ' of', '4' => ' is');

my $line1 = 'MEDLINE=97105885; PubMed=4248633; DOI=10.1093/nar/24.22.4420;';
my $line2 = '55AAB+33AAC-4EEE+343+GHT--2CBA+6--4ABC-456';
my $line3 = '   123456 This Perl course is almost over 654321   ';

my $res = '';

if ( $line1 =~ /\s\w{3,7}=(\d{2})\.?\w{5}/ ) {
    $res .= $1;
}

my $cnt = 0;
while ($line2 =~ /[123][ABC]{3}(\+|-)[456]/g) {
   $cnt++;
}
$res .= $tr{$cnt};

$line3 =~ s/^\s+\d+//;
$line3 =~ s/\d+\s+$//;
$line3 =~ s/(\s+\w+){3}//;
$line3 =~ s/(\s+\w+).*$/$1/;
$res .= $line3;

print "$res\n";
```

Note: \w matches the following characters [a-zA-Z0-9_]

```
1  42 it is
```

Question 4 (2p): What is the output of the following program?

```
#! /usr/bin/perl -w
use strict;

my @arr1 = ('Start', -2, 5, 2, 10, 40, -6, 4, 7, -25, 8);
shift @arr1;

my @arr2;
foreach my $item (@arr1) {
   push @arr2, $item if ($item > 0);
}

my @arr3 = sort {$a <=> $b} @arr2;

my $sum = (shift @arr3) + (pop @arr3);

print "$sum\n";
```

```
1 42
```

Question 5 (2p): What is the output of the following program?

```
#! /usr/bin/perl -w
use strict;

my @delims = sort {$a cmp $b} ('t', 'g', 'a', 'c');
my $delim = pop @delims;
my $rm = shift @delims;

my $a = 'eaatantittaaf';

$a =~ s/$rm//g;
my @res = split /$delim+/, reverse($a);
my $ans = join ' ', @res;

print $ans, "\n";
```

```
1 f i n e
```

Part 2: Write Perl code

Question 6 (3p): Your scalar `$text` contains text (some sentences, a written report or perhaps even a book). We can assume that `$text` is free of linebreaks. Write **Perl code** that counts the number of sentences that starts with the word "The". Note: You can assume that a sentence ends with either "." or "!".

```
#! /usr/bin/perl -w
use strict;

# Not actually part of the solution
my $text = 'The is a word! I like Perl! It is The language. The linux system needs Perl. There you are!';

my $cnt = 0;
while ( $text =~ /(^The|(\.|!)\s+The\s)/g ) {
    $cnt++;
}
print "Found $cnt\n";

# Another solution using split
$cnt = 0;
my @parts = split /!|\./, $text;
foreach (@parts) {
   $cnt++ if (/^\s*The\s/)
}
print "Found $cnt\n";
```

Question 7 (3p): The unix command `head` prints the first 10 lines of a given file. That is, if you type on your command line,

```
>> head convert.pl
```

then you will see the first 10 lines of the file convert.pl. Write a **Perl program** that works like the unix command `head`. Note, the program should work also for files containing less than 10 lines.

```
#! /usr/bin/perl -w
use strict;

# Robust solution
unless ( defined($ARGV[0]) ) {
    die "No file specified\n";
}
open my $file, '<', "$ARGV[0]" or die "Cannot open file $ARGV[0]. $!\n";
my $cnt = 0;
while ( my $line = <$file> ) {
    print $line;
    $cnt++;
    last if ( $cnt == 10 );
}
close($file);

# Shorter solution
```

```
my $cnt = 0;
while ( my $line = <> ) {
    print $line;
    $cnt++;
    last if ( $cnt == 10 );
}

# Even shorter!
while ( <> ) {
    print;
    last if ( $. == 10 );
}
```

Question 8 (3p): Write **Perl code** that computes the sum of all odd values for 1 up to 999.

```
#! /usr/bin/perl -w
use strict;

my $sum = 0;
for( my $odd = 1; $odd <= 999; $odd += 2 ) {
    $sum += $odd;
}
print "The sum is $sum\n";

# Another solution, slower though
my $sum2 = 0;
foreach my $i (1..999) {
    $sum2 += $i unless( $i%2 == 0 );
}
print "The sum is $sum2\n";

# Or the math solution :-)
print "The sum is ", 1000 * int(999/2) / 2 + int(999/2)+1, "\n";

```

Question 9 (3p): Write a **subroutine** that takes references to two arrays as arguments and returns a reference to a hash. The hash is created by taking the items of the first array as keys and items of the second array as the corresponding values. The number of keys-values pairs in the hash should be the number of items in the shortest of the two arrays.

```
#! /usr/bin/perl -w
use strict;

my @arr1 = qw/hello world TAA Five/;
my @arr2 = qw/1 2 3 4 5 6/;

my $res = mkhash(\@arr1, \@arr2);
while (my ($key, $value) = each %{$res}) {
    print "$key - $value\n";
}

```

```
12
13 sub mkhash {
14     my($arr1, $arr2) = @_;
15
16     my $items = (@{$arr1} < @{$arr2} ? @{$arr1} : @{$arr2});
17
18     my %hash;
19     for( my $i = 0; $i < $items; ++$i ) {
20         $hash{$arr1->[$i]} = $arr2->[$i];
21     }
22
23     return \%hash;
24 }
```

Question 10 (3p): Write a **Perl program** that reads all arguments given on the commande line. The program should print all the arguments (one argument per line) containing only nucleotide sequences. E.g.

```
>> ./q10.pl Hello ACGTTGAAC world 23 ACGT 123TTAA TATATATA
```

would respond with

```
ACGTTGAAC
ACGT
TATATATA
```

```
1 #! /usr/bin/perl -w
2 use strict;
3
4 foreach my $arg (@ARGV) {
5    print "$arg\n" if ($arg =~ /^[ACGT]+$/);
6 }
7
```

Good Luck! /Mattias