YPH 205
SOLUTIONS TO PRESCRIBED PERL PROGRAMS

This document was typeset using LaTeX. If you are interested, you can download and examine the `.tex` file at `vhbelvadi.com/teaching`, suggest edits and alternate solutions and more. Updated 09.02.2020.

EXECUTION INSTRUCTIONS FOR ALL PROGRAMS

1. Open Terminal, Command Prompt or equivalent Command Line Interface.
2. Create a new folder with `mkdir <folderName>`. **Use new folders for each program.**
3. Open the new folder with `cd <folderName>` and type `vi <name>.pl` to open a new file.
4. Type your program in the file. Use `I` to start typing and `Esc` to stop typing.
5. After typing, hit `Esc` then save the program with `:w` and quit with `:q`, or just use `:wq`.
6. Run the program with `perl <name>.pl`.

In all cases above, replace `<...>` *with a name of your choice (obviously). Avoid spaces in file names.*

1. SEARCHING FOR A PATTERN IN A STRING

SPECIAL INSTRUCTIONS FOR THIS PROGRAM

1. Create a text file `vi input.txt` with three lines of data e.g.:
   This is a sample
   text file used
   for PERL programs.
2. Create a perl file `vi pattern-match.pl` and type the program below.

```
1   #! /usr/bin/perl -w
2
3   open (FILE, 'input.txt') or die "$!";
4   while (<FILE>) {
5           if (m/text file/) {
6                   for (1..10) {
7                           <FILE>;
8                   }
9                   print;
10                  last;
11          }
12  }
13  close FILE;
```

2. COUNTING LINES, WORDS AND CHARACTERS IN A FILE

SPECIAL INSTRUCTIONS FOR THIS PROGRAM

Use the same input file as in the previous program.

```
#!/usr/bin/perl -w
open(FILE, "textfile.txt") or die "Could not open file: $!";

my ($lines, $words, $chars) = (0,0,0);

while (<FILE>) {
    $linecount++;
    $charcount += length($_);
    $wordcount += scalar(split(/\s+/, $_));
}

print ("Total number of lines: $linecount \n");
print ("Total number of words: $wordcount \n");
print ("Total number of characters: $charcount \n");
```

3. SORTING STRINGS

```
#!/usr/bin/perl -w
my @strings = qw(quantum relativistic classical);
my @sorted = sort @strings;
print "\nSorted strings:\n";
print join "\n", @sorted;
```

4. CHECKING FOR PRIME NUMBERS

```
#! /usr/bin/perl -w
print "Enter a number:\t";
$number = <>;
$divisor = 0;
$flag = 0;
if ($number != 2) {
        for ($divisor = 2; $divisor < $number ; $divisor++) {
```

```
40                  if ($number % $divisor == 0) {
41                          $flag = 1;
42                          last;
43                  }
44          }
45  } else {
46          $flag = 1;
47  }
48  if ($flag != 1) {
49          print "The number is prime.\n";
50  } else {
51          print "The number is composite.\n";
52  }
```

5. FINDING THE ROOTS OF A QUADRATIC EQUATION

```
53  #! /usr/bin/perl -w
54  use Math::Complex;
55  INPUT:
56  print "\nEnter the three non-zero co-efficients of ax^2 + bx + c:\n";
57  chomp($a=<>, $b=<>, $c=<>);
58  if ($a == 0) {
59          print "The co-efficient a cannot be zero. Try again.\n";
60          goto INPUT;
61  }
62  ($x1, $x2) = solveQuad($a,$b,$c);
63  print "Root 1 = $x1, Root 2 = $x2\n\n";
64  sub solveQuad {
65          my ($a, $b, $c) = @_;
66          my $root = sqrt($b**2 - 4*$a*$c);
67          return (-$b + $root)/(2*$a), (-$b-$root)/(2*$a);
68  }
```

6. LEAST SQUARE FITTING DATA FROM A FILE

SPECIAL INSTRUCTIONS FOR THIS PROGRAM

Create a tab-separated input file with two columns (x and y) of numbers.

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

print "\nEnter the name of the input file:\t";
$file = <>;
print "Enter the number of rows of data:\t";
$rows = <>;

open (INPUT,$file) or die "Could not open the file";
# @line;
@arrayX = ();
@arrayY = ();
$counter = 0;

($i, $sumx, $sumy, $sumxy, $sumx2) = (0, 0, 0, 0, 0);

while(<INPUT>){
    @line = split(/\t/,$_);
    $arrayX[$counter]=$line[0];
    $arrayY[$counter]=$line[1];
    $counter++;
}

for($i=0;$i<$rows;$i++)
    {
        $sumx = $sumx + $arrayX[$i];
        $sumx2 = $sumx2 + $arrayX[$i]*$arrayX[$i];
        $sumy = $sumy + $arrayY[$i];
        $sumxy = $sumxy + $arrayX[$i]*$arrayY[$i];

    }

$c = (($sumx2*$sumy - $sumx*$sumxy)*1.0/($rows*$sumx2-$sumx*$sumx)*1.0);
$m = (($rows*$sumxy-$sumx*$sumy)*1.0/($rows*$sumx2-$sumx*$sumx)*1.0);
print "\n\nThe line of best fit is y=$m x + $c \n\n";
```