CS 2413 Test 1

1. [10] Write a sed command which will take a data file, `data.in` containing only numbers and strip off the first number of each line, writing the result to the file `data.out`. There may or may not be white space before the first number and all of the white space after the first number should be stripped. The sed command should be entirely on the command line.

2. [15] You are running `portsentry` which will detect tcp and udp connection attempts at a number of ports and will block connections from those sites. You want a list of all TCP and UDP connection attempts not to TCP port 80 from `scanner.cs.utsa.edu` and since they are logged in `/var/log/messages` you want a sed script to print out the date, `scanner.cs.utsa.edu`, the connection type (TCP or UDP) and the port. If the input looks like:

```
Mar 5 09:16:56 attackalert: Connect from: scanner.cs.utsa.edu to TCP port: 635
Mar 5 10:29:21 adminalert: Going into listen mode on TCP port: 4000
Mar 5 12:36:18 attackalert: Connect from: scanner.cs.utsa.edu to ~~TCP port: 80~~
Mar 5 16:32:31 attackalert: Connect from: 129.115.11.71 to UDP port: 137
Mar 5 16:35:13 attackalert: Connect from: scanner.cs.utsa.edu to UDP port: 137
Mar 5 19:17:09 attackalert: Connect from: 129.115.11.57 to TCP port: 1080
Mar 5 19:17:26 attackalert: Connect from: ten43.cs.utsa.edu to TCP port: 79
```

then the output should look like:

```
Mar 5 09:16:56 scanner.cs.utsa.edu TCP 635
Mar 5 16:35:13 scanner.cs.utsa.edu UDP 137
```

Notice that only lines with `attackalert` have information to be printed. Write the sed script!

3. [15] The `last` command prints out the connection times of the users of the system in a form similar to the following:

use awk

```
 #1       #2      #3              #4  #5  #6  #7  #8  #9
maynard   ftp    129.115.11.102   Wed Oct 11 9:10 - 10:15  65 minutes
grader    ftp    129.115.176.200  Wed Oct 11 10:18 - 1:18  180 minutes
maynard  pts/9   129.115.203.145  Sun Oct 15 10:13 - 20:22  609 minutes
george  pts/9   129.115.203.145  Mon Oct 16 2:03 - 2:23  20 minutes
```

You are curious about the activity of the user maynard connecting via ftp from the site 129.115.203.145 as well as other users who connecting via ftp from that site. In particular you are interested in:

(a) The number of times and total time that **maynard** was connected from 129.115.203.145 on October 15 via ftp.

(b) The number of times and total time that all users were connected from 129.115.203.145 on October 15 via ftp.

(c) The number of times and total time that all users were connected from any site on October 15 via ftp.

The output should look like:

```
maynard      129.115.203.145     120 connections      427 minutes
all users    129.115.203.145     183 connections      613 minutes
all users    all sites           474 connections     2498 minutes
```

4. [15] Write a perl script which will print out the pathnames of all text files which are under any of the directories whose names are given in the files on the command line.

5. [15] Write a Perl script, `pil`, which will interleave the lines of a file with those of another file writing the result to a third file. If the files are a different length then the excess lines are written at the end. A sample invocation:

```
pil filein1 filein2 fileout
```

6. [15] Write a perl script which will print on stdout the names of all `text` files under any of the directories given on the command line which contain the string "sniffer". A file name should only be printed a maximum of one time.

7. [20] Suppose a perl program, `tct`, is invoked with the command:

```
tct -xt file1 file2
```

and suppose `file1` contains the lines "`dir1`" and "`dir2`" while `file2` contains the lines "`dir3`", "`dir4`", "`dir5`" and "`dir6`". Each string is on a separate line in the files. At the start of the program:

(a) What is the list value of `@ARGV`? [-xt file1 fi2

(b) What is the value of `$ARGV[2]`? file 2

(c) What is the value of `$#ARGV`? 2

(d) What value is used for `@ARGV` in the condition test:

```
    if ( @ARGV ) {
```

CS 2413 Test 1 Solutions

1. [10]

```
sed 's/^[  ]*[0-9][0-9]*[  ]*//' data.in > data.out
```

2. [15] Just delete the lines and fields that you don't want.

```
/TCP port: 80/d
/attackalert/!d
/scanner\.cs\.utsa\.edu/!d
s/attackalert: Connect from: //
s/to //
s/ port://
```

3. [15] Nesting the conditions will yield cleaner and shorter code.

```
BEGIN{
  sconn = 0; stime = 0; aconn = 0; atime = 0; mconn = 0; mtime = 0
}
$5 == "Oct" && $6 == 15 && $2 == "ftp" {
   sconn++
   stime += $10
   if ( $3 == "129.115.203.145" ) {
     aconn++
     atime += $10;
     if ( $1 == "maynard" ) {
       mconn++
       mtime += $10
     }
   }
}
END{
  print "maynard\t129.115.203.145\t" mconn,"connections\t" mtime,"minutes"
  print "all users\t129.115.203.145\t" sconn,"connections\t" stime,"minutes"
  print "all users\tall sites\t" aconn,"connections\t" atime,"minutes"
}
```

4. [15]
```
@dirs = <>;
chop @dirs;
@files = `find @dirs -print`;
chop @files;
@tfiles = grep(-f && -T, @files);
print join("\n", @tfiles),"\n";
```

5. [15] The condition is important here. Because of short circuiting you cannot read as part of the condition. The `||` works since printing the null string has no effect. If an `&&` is used then separate code is necessary to print any extra lines from either filehandle.

```
open(IN1,$ARGV[0]) || die "Unable to open $ARGV[0]:$!\n";
open(IN2,$ARGV[1]) || die "Unable to open $ARGV[1]:$!\n";
open(OUT,">$ARGV[2]") || die "Unable to open $ARGV[2]:$!\n";
$line1 = <IN1>;
$line2 = <IN2>;
while ( $line1 || $line2 ) {
  print OUT $line1, $line2;
  $line1 = <IN1>;
  $line2 = <IN2>;
}
```

This can be done using lists in the following manner:

```
open(IN1,$ARGV[0]) || die "Unable to open $ARGV[0]:$!\n";
open(IN2,$ARGV[1]) || die "Unable to open $ARGV[1]:$!\n";
open(OUT,">$ARGV[2]") || die "Unable to open $ARGV[2]:$!\n";
@lines1 = <IN1>;
@lines2 = <IN2>;
while ( @lines1 || @lines2 ) {
  print OUT shift @lines1, shift @lines2;
}
```

6. [15]

```
@files = `find @ARGV -print`;
chop @files;
@tfiles = grep(-f && -T, @files);
FILE:
foreach $file (@tfiles) {
  open(FILE,$file) || next;
  while ( <FILE> ) {
    if ( /sniffer/ ) {
      print $file,"\n";
      next FILE;
    }
  }
}
```

7. [20]

```
(a) @ARGV == ('-xt', 'file1', 'file2')
(b) $ARGV[2] == 'file2'
(c) $#ARGV == 2
(d) scalar @ARGV == 3
(e)
    i.   $a == 'file2'
    ii.  $c == "dir1\n"
    iii. $ARGV == 'file1'
(f)
    i.   $a == "-xt"
    ii.  @b == ("dir2\n", "dir3\n", "dir4\n","dir5\n","dir6\n")
    iii. $c == "dir1\n"
```