CS 2413 Test 1

1. **[10]** Write a sed command which will add two zeros to the front of each 5 digit integer occuring at the end of a line. The input data should be read from the file `mp3db` and writing the output to `mp3db.out`. The command should be entirely on the command line.

2. **[15]** There is a new "attack" on systems running ssh where there are attempted logins as a number of different users. Since this is not a mechanism that should compromise many systems, you are curious to find a system with the exploit so that you can determine how it was compromised and whether the attack is "hiding" a more sophisticated attack. Write a sed script to get the sending ips from a number of different log files. The lines from which you want to take the ips look like:

```
Oct 17 08:41:40 sylph sshd[8083]: Illegal user test from ::ffff:212.168.26.139
Oct 17 08:41:43 sylph sshd[8084]: Illegal user guest from ::ffff:72.68.127.15
```

   There are many other lines in this file so you only want to match these line for any illegal user who connected via sshd. The above lines should cause the following output and should work for any illegal user:

```
212.168.26.139
72.68.127.15
```

   Be careful not to get ips from other lines. Other lines that might appear are:

```
Oct 17 11:16:49 sylph portsentry[15291]: attackalert: Blocking 207.228.236.8
Oct 17 08:41:40 sylph sshd[8083]: input_userauth_request: illegal user test
Oct 17 08:41:43 sylph sshd[8084]: Failed password for illegal user guest from
::ffff:72.68.127.15 port 40730 ssh2
```

3. **[15]** An ISP (Internet Service Provider) is concerned about a user named maynard spamming email so they want a tool to look in the syslog file and pull out information about maynard's email usage on March 7. Write a awk script which will determine the total size of the outgoing emails **written by maynard** as well as the total number of email messages **received by maynard** on **March 7**. The input data will include several months worth of data for many users. The key lines (simplified from the real syslog data) look like:

```
Apr  5 21:58:18 medusa sendmail[6432]: to=<smaynard@medusa>, stat=Sent
Mar  7 00:06:37 medusa sendmail[7248]: from=<maynard@medusa>, size = 681
Mar 15 19:28:13 medusa sendmail[6432]: to=<maynard@medusa>, stat=Sent
```

   The output should look like:

```
maynard
    1573889 bytes mailed out
    798 emails received
```

4. **[10]** Write a Perl script which will print out the pathnames of all directories which are under any of the directories which are given in the files whose names are given on the command line.

5. **[15]** Write a Perl script, called `psplit`, which, given a number of lines, will break a file into a number of files each of which have the given number of lines except, possibly, the last file. The files will have a period followed by the index number appended to the end of the original file name. Thus if file.doc has 345 lines, the command
`psplit 100 file.doc`
would create the files `file.doc.1`, `file.doc.2`, and `file.doc.3` each with 100 lines and a final file, `file.doc.4` which will only have 45 lines.

6. **[20]** Write a Perl script, `pgrep`, which will search all **TEXT** files under the directories given on the command line for the regular expression given as the first argument on the command line. Print out the pathname of each file followed by a colon and the matching line for each line in which a match is found. A sample invocation:
`pgrep '\w+ [mM]aynard' file1 file2 file3`

7. **[20]** Suppose a perl program, `backup`, is invoked with the command:
`backup -ap -f sys1 sys2 sys3`
and suppose the file `sys1` contains the lines "`dir1`" and "`dir2`" while the file `sys2` contains the lines "`dir3`", "`dir4`", "`dir5`" and "`dir6`" and the file `sys3` contains "`dir7`". At the start of the program:

   (a) What is the list value of `@ARGV`?
   (b) What is the value of `$ARGV[2]`?
   (c) What is the value of `$#ARGV`?
   (d) What value is used in the condition test:
   `print "Usage: back [-apf] <file system lists>\n" if @ARGV < 2;`
   (e) If the following statements are executed first,

```
$a = shift;
$b = pop;
$c = shift;
$d = shift;
push(@ARGV,$d);
$e = <>;
```

      i. What is the value of `$c`?
      ii. What is the value of `$e`?
      iii. What is the value of `$ARGV`?

   (f) On the other hand if the following statements were to be executed first,

```
$a = shift;
$b = shift;
$c = shift;
@c = <>;
$d = pop @c;
```

      i. What is the value of `$c`?
      ii. What is the value of `@c`?
      iii. What is the value of `$d`?

Fall '04

CS 2413 Solutions to Test 1

1. [10]

```
sed '/[^0-9][0-9][0-9][0-9][0-9][0-9]$/s/$/00/' < mp3db > mp3db.out
```

or for extra credit [15]

```
s/\([^0-9]\)\([0-9][0-9][0-9][0-9][0-9]\)$/\100\2/ < mp3db > mp3db.out
```

2. [15]

```
/sshd/!d
/Illegal user/!d
/:[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*$/!d
s/^.*ffff://
```

3. [15]

```
BEGIN{ rec=0; sent=0 }
/Mar 7 / && $6 == "to=<maynard@medusa>," && $7 == "stat=Sent" {
  rec++
}
/Mar 7 / && $6 == "from=<maynard@medusa>," && $7 == "size" {
  sent += $9
}
END{
    print "maynard"
    print "\t" sent,"bytes mailed out"
    print "\t" rec,"emails received"
}
```

4. [10]

```
@dirs = <ARGV>;
chop @dirs;
@files=`find @dirs -print`;
chop @files;
@dirs = grep(-d , @files);
print join("\n",@dirs),"\n";
```

5. [15]

```
$num = shift;
@lines = <ARGV>;
$n = 1;
while ( @lines ) {
  open(OUT,"> $ARGV[0].$n") or die "Unable to open $ARGV[0].$n: $!\n";
  if ( $num <= @lines ) {
    print @lines[0..($num-1)];
    @lines = @lines[$num..$#lines];
  } else {
    print @lines[0..$#lines];
    last;
  }
  $n++;
}
```

6. [20]

```
$rexp = shift;
@files = `find @ARGV -print`;
chop @files;
@tfiles = grep( -f && -T, @files);
foreach $file (@files) {
  open(FILE,$file) or next;
  while ( <FILE> ) {
    if ( /$rexp/ ) {
      print "$file:$_" if /$rex/;
      next FILE;
    }
  }
}
```

7. [20]

(a) @ARGV == ("-ap", "-f", "sys1", "sys2", "sys3")
(b) $ARGV[2] == "sys1"
(c) $#ARGV == 4
(d) scalar @ARGV == 5
(e)
    i. $b == "-f"
    ii. $d == "dir3\n"
    iii. $ARGV == "sys2"
(f)
    i. $b == "sys1"
    ii. @b == ("dir3\n", "dir4\n", "dir5\n", "dir6\n")
    iii. $c == "dir7\n"