CS 3423 Test 1

1. [10] Write a `sed` command to be executed on the command line which will read a data file, `data.in`. Some of the lines have an alphabetic code (upper/lower case alphabetic characters) as the last token on the line while the remaining lines have a number as a last token or are blank. Print out every alphabetic code, one per line with no blank lines, writing the output to the file `codes.dat`. The printed code should not contain white space.

2. [15] Being annoyed by the constant bogus ssh login attempts looking for stupid passwords which clog your log files, you have written a program which rejects the route back to the scanner after 5 failed login attempts in a row. Now you want to write a `sed` script which will restore the rejected routes. Your script will process the output of the `route -n` command which looks like:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
209.196.35.226  -               255.255.255.255 !H    0      -        0 -
60.173.26.12    -               255.255.255.255 !H    0      -        0 -
129.115.27.0    0.0.0.0         255.255.255.0   U     10     0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     10     0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         129.115.27.253  0.0.0.0         UG    10     0        0 eth0
```

The rejected routes have a "`!H`" as the Flag (4^{th} token) and you want to delete these rejected routes. The commands to delete the two rejected routes above (which you are to print) are:

```
/usr/sbin/route del -host 209.196.35.226 reject
/usr/sbin/route del -host 60.173.26.12 reject
```

3. [15] Since you have been seeing a number of these attacks comming from UTSA (129.115) you want to know how many come from the various CS domains as well as the rest of UTSA. Using the data format from the preceeding problem, write an `awk` script which will count the number of rejected routes from `129.115.27`, `129.115.28`, `129.115.29`, `129.115.30`, the rest of UTSA as well as the total number of rejected routes and show the attacking ips for the CS domains under the correct domain. For 10 points you can simply print the totals.

```
129.115.27              3
        129.115.27.13
        129.115.27.48
        129.115.27.147
129.115.28              1
        129.115.28.134
129.115.29              2
        129.115.29.212
        129.115.29.213
129.115.30              0
Other UTSA             28
Total Rejected Routes  79
```

4. [10] Write a Perl script which will find and print the pathnames of all NONREGULAR files which are UNDER the directories whose names are on the command line.

5. [15] Write a Perl script, `extract`, which will write the last 100 lines in each file given on the command line into the output file in the same order as the filenames on the command line. If a file does not have 100 lines then the entire file should be written into the output file. There may be any number of input files. Unfortunately all of the input files may not fit in memory at the same time so this needs to be taken into consideration.

```
extract filein1 filein2 filein3 filein4 fileout
```

6. [20] Write a Perl script, `pgrep`, which will search all **TEXT** files under the directories whose names are in the files whose names are on the command line for the regular expression given as the first argument to `pgrep`. For each line which contains a match the pathname of the file should be printed followed by a colon and then the matching line. A sample invocation:
`pgrep '\w+ca*t' file1 file2 file3 file4`

7. [20]Suppose a perl program, `logrotate`, is invoked with the command:
`logrotate -l message syslog security users`
and suppose `message` contains the lines "`message`" and "`message1.gz`" while `syslog` contains the lines "`syslog`", "`syslog1.gz`" and "`syslog2.gz`" and `security` contains the lines "`security.log`" and "`security1.log.gz`" and the file `users` contains the line "`users.log`". Each string is on a separate line in the files. At the start of the program:

   (a) What is the list value of `@ARGV`?
   (b) What is the value of `$ARGV[1]`?
   (c) What is the value of `$#ARGV`?
   (d) What value is used for `@ARGV` in the condition test:

```
if ( @ARGV > 1 )
```

   (e) After the following statements are executed,

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

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

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

```
$a = pop;
@b = splice(@ARGV,0,3,($a));
@c = <ARGV>;
```

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

CS ~~2413~~ 3423 Solutions to Test 1

1. [10]

```
sed '/[a-zA-Z]$/!d;s/^.*[      ]//' data.in > codes.dat
```

2. [15]

```
/!H/!d
s/[     ].*$/ reject/
s/^/\/usr\/sbin\/route del -host /
```

Note: If this were on the command line then ! would need to be escaped.

3. [15]

```
BEGIN{n27=n28=n29=n30=nutsa=nall=0}
/^129\.115\.27\./ && /!H/{
  n27++
  ssh27[$1] = 1
}
/^129\.115\.28\./ && /!H/{
  n28++
  ssh28[$1] = 1
}
/^129\.115\.29\./ && /!H/{
  n29++
  ssh29[$1] = 1
}
/^129\.115\.30\./ && /!H/{
  n30++
  ssh30[$1] = 1
}
/^129\.115\./ && /!H/{
  nutsa++
}
/!H/ {
  nall++
}
```

```
END {
      print "129.115.27\t\t" n27
      for (ip in ssh27) {
        print "\t" ip
      }
      print "129.115.28\t\t" n28
      for (ip in ssh28) {
        print "\t" ip
      }
      print "129.115.29\t\t" n29
      for (ip in ssh29) {
        print "\t" ip
      }
      print "129.115.30\t\t" n30
      for (ip in ssh30) {
        print "\t" ip
      }
      print "Other UTSA\t\t" nutsa-n27-n28-n29-n30
      print "Total Rejected Routes\t" nall
}
```

Note: $4 == "!H" is more accurate than /!H/ which is only acceptable because of knowledge of the output of the route command.

4. [10]

```
@files = `find @ARGV -print`;
chop @files;
@nfiles = grep(!-f, @files);
print join("\n", @nfiles),"\n";
```

5. [15]

```
$outfile = pop(@ARGV);
open(OUT,">$outfile") or die "Unable to open $outfile:$!\n";
FILE:
foreach $file (@ARGV) {
  open(FILE,$file) or next FILE;
  @lines = <FILE>;
  if ( @lines >= 100 ) {
    print OUT @lines[$#lines-99..$#lines];
  } else {
    print OUT @lines;
  }
}
```

6. [20]

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

7. [20]

(a) @ARGV == ('-l', 'message', 'syslog', 'security', 'users')
(b) $ARGV[1] == 'message'
(c) $#ARGV == 4
(d) scalar @ARGV == 5
(e)
    i. $b == 'message'
    ii. $e == "syslog1.gz\n"
    iii. $ARGV == 'syslog'
(f)
    i. $a == "users"
    ii. @b == ('-l', 'message', 'syslog')
    iii. @c == ("users.log\n", "security.log\n", "security1.log.gz\n")