CS 2413 Test 1

1. **[10]** Write a **sed command** which will change every maximal sequence of white space (blanks and tabs) to a single "*". The input should come from the file `text.in` and the output should be written to the file `text.out`. The command should be entirely on the command line.

2. **[15]** There are a large number of machines which are performing TCP SYN scans of your mail server on `sylph` at port 25 in order to locate machines which can be used to relay spam. You are running portsentry which will reject their route so that they cannot communicate with `sylph` but you have decided that you want to create a list of the ips which are doing this scanning. As a first step you want to write a **sed script** which will locate the appropriate lines and reduce the lines to merely the ip address. A line indicating a TCP SYN scan directed at `sylph` to port 25 looks like

```
Mar 20 17:13:22 sylph: TCP SYN scan from 84.165.94.9 to TCP port: 25
```

There are other lines which should not be processed because they are not directed at sylph or are not directed at TCP port 25 or are not a TCP SYN scan. Of the following lines only the first should result in any output:

```
Mar 20 17:13:22 sylph: TCP SYN scan from 84.165.94.9 to TCP port: 25
Mar 20 17:13:23 sylph: Host: 84.165.94.9 is already blocked Ignoring
Mar 20 17:47:34 dryad: TCP SYN scan from 184.165.94.9 to TCP port: 25
Mar 20 17:53:11 sylph: TCP SYN scan from 4.65.9.123 to UDP port: 165
Mar 20 17:56:32 sylph: TCP FIN scan from 84.165.94.9 to TCP port: 25
Mar 20 17:59:51 sylph: TCP SYN scan from 133.67.21.13 to UDP port: 25
```

and this output should be

```
84.165.94.9
```

3. **[15]** You know that a number of the MS machines at your site have been infected with the Welchia worm which is performing a CyberKit scan of other machines. In your log file a Cyberkit scan alert and other alerts looks like the following where the number of scans of a given type is given at the end of each line:

```
CyberKit 2.2 129.115.66.11 -> 129.115.30.31 9 times
WEB-IIS cmd.exe access 142.12.14.4:3258 -> 129.115.30.31:80 2 times
CyberKit 2.2 129.115.66.12 -> 129.115.30.30 3 times
WEB-IIS cmd.exe access 145.15.54.4:5251 -> 129.115.30.30:80 3 times
SCAN SOCKS Proxy attempt 81.53.34.62:3000 -> 129.115.30.11:1080 4 times
```

   Using the above data, write an **awk** script which will count the total number times an attack (or scan) was recognized, the total number of CyberKit scans and the total number of WEB-IIS attacks. In addition determine how many of these were specifically directed towards 129.115.30.31. Your output should look like:

```
CyberKit scans at 129.115.30.31      9
WEB-IIS attacks at 129.115.30.31     2
Total attacks at 129.115.30.31      11
Total CyberKit scans                12
Total WEB-IIS attacks                5
Total attacks                       21
```

4. **[10]** Write a Perl script which will take a list of names of files and directories on the command line and print out the pathnames of all regular non-text (binary) files which are either one of the files on the command line or are under any of the directories given on the command line.

5. **[15]** Write a Perl script, called `fmerge`, which is given two input filenames on the command line and an output filename. `fmerge` should append each line of the second input file to the end of the corresponding line in the first file separated by a blank, writing the new line to the output file. Thus if line 4 of first input file is `"cat\n"` and line 4 of the second input file is `"dog\n"`, line 4 of the output file should be `"cat dog\n"`. The files need not be the same length, missing lines should be handled as if they were null strings (be careful of the newline character in this case).

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 given on the command line for the regular expression given as the first argument on the command line. For the first match in each file, print out the pathname of the file followed by a colon and then the matching line. Pay careful attention to the fact that a pathname will occur a maximum of **one** time. A sample invocation:
   `pgrep '\w+ca*t' file1 file2 file3 file4 file5 file6`

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

   `snoop -s -x cs.machines bio.machines math.machines`

   and suppose the file `cs.machines` contains the single line "`ten01`", the file `bio.machines` contains the lines "`bio01`", "`bio02`" and "`bio03`" while the file `math.machines` contains "`mat01`" and "`mat02`". At the start of the program:

   (a) What is the list value of `@ARGV`?

   (b) What is the value of `$ARGV[3]`?

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

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

   ```
   print "Usage: snoop [-s] [-x] <machine file list>\n" unless @ARGV > 0;
   ```

   (e) If the following statements are executed first.

   ```
   $a = pop;  math
   $b = pop;  bio
   unshift(@ARGV,$b);  bio   -s
   $c = <>;
   ```

   i. What is the final value of `$b`?

   ii. What is the final value of `$c`?

   iii. What is the final value of `$ARGV`?

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

   ```
   $a = pop;  math
   $b = shift;  -s
   $b = shift;  -x
   @b = <>;  ten01
   $c = pop @b;  ten01
   ```

   i. What is the final value of `$b`?

   ii. What is the final value of `@b`?

   iii. What is the final value of `$c`?

CS 2413 Solutions to Test 1

1. **[10]**

```
sed 's/[   ][   ]*/*/g' < text.in > text.out
```

2. **[15]**

```
/ sylph /!d
/ TCP SYN /!d
/ TCP port: 25$/!d
s/^.*from //
s/ to .*$//
```

3. **[15]**

```
BEGIN{ ck=0; tck=0; web=0; tweb=0; t31=0; total=0 }
/^CyberKit /{ tck += $5
              if ( $4 == "129.115.30.31" ) ck += $5
            }
/^WEB-IIS /{ tweb += $6
             if ( $5 == "129.115.30.31:80" ) web += $6
           }
/-> 129\.115\.30\.31[ :]/ { t31 += ${NF-1} }
           { total += ${NF-1}
           }
END{
  print "CyberKit scans at 129.115.30.31" "\t" ck
  print "WEB-IIS attacks at 129.115.30.31" "\t" web
  print "Total attacks at 129.115.30.31" "\t" t31
  print "Total CyberKit scans" "\t" tck
  print "Total WEB-IIS attacks" "\t" tweb
  print "Total attacks" "\t\t\t" total
}
```

4. **[10]**

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

5. **[15]**

```
open(IN1,$ARGV[0]) or die "Unable to open $ARGV[0] for read:$!\n";
open(IN2,$ARGV[1]) or die "Unable to open $ARGV[1] for read:$!\n";
open(OUT,">$ARGV[2]") or die "Unable to open $ARGV[2] for write:$!\n";
@in1 = <IN1>;
chop @in1;
@in2 = <IN2>;
chop @in2;
while ( @in1 || @in2 ) {
  print OUT shift @in1, " ", shift @in2, "\n";
}
```

6. **[20]**

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

7. **[20]**

```
(a) @ARGV == ("-s", "-x", "cs.machines", "bio.machines", "math.machines")
(b) $ARGV[3] == "bio.machines"
(c) $#ARGV == 4
(d) scalar @ARGV == 5
(e)
    i.   $b == "bio.machines"
    ii.  $d == "bio01\n"
    iii. $ARGV == "bio.machines"
(f)
    i.   $b == "-x"
    ii.  @b == ("ten01\n", "bio01\n", "bio02\n")
    iii. $c == "bio03\n"
```