CS 2413 Test 1

1. [10] Write a sed command which will delete all **blank** lines that are **not** null lines in the file **in.data** and pipe the output into the program **wc**. The command should be entirely on the command line.

2. [15] Write a sed script which will print out a list of the names of the included files. Only the name of the included files should appear, one per line. Included files appear in one of the following two forms:

```
#include <filename1.h>
#include "filename2.h"
```

which would cause the following output

```
filename1.h
filename2.h
```

3. [15] An ISP (Internet Service Provider) has a data file which contains a month's worth of information about user's connect times and bandwidth usage as well as storage usage. There are two types of lines in the file, a connect line for each user connection which has 5 fields containing the login, start and stop times as well as the bandwidth usage broken into the number of bytes in versus the number of bytes out.

```
<user name> <start time> <stop time> <bytes in> <bytes out>
maynard        1378          2463        12379894     34563
```

and daily storage usage lines which contains the number of bytes of disk space being used by the user which has the following format:

```
*storage*  <user name> <bytes of storage>
*storage*    maynard         2784643
```

Write an awk script to find whether the user **maynard** has a total connect time exceeding 10,000 or whose total bandwidth consumed (both in and out) exceeds 100,000,000,000 bytes or whose maximum storage for the month exceeded 10,000,000. If any of these conditions are valid, then print out all statistics for the user **maynard**. The output should look like:

```
maynard
        Connect Time = 1085
        Bandwidth    = 12314457
        Storage      = 2784643
```

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

5. [20] Suppose a perl program, pargs, is invoked with the command:

```
pargs -xvf 'a+ b+' file1 file2 file3
```

and suppose file1 contains "dir1", "dir2" and "dir3" while file2 contains "dir4" and "dir5" and file3 contains "dir6" and "dir7". Each directory name in the files is on a separate line. At the start of the program:

(a) What is the 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:

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

(e) If the following statements are executed first,

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

i. What is the value of $c[1]?

ii. What is the value of @c?

iii. What is the value of $d?

6. [15] Write a Perl script, pgrep, which will search all **TEXT** files given on the command line for the regular expression given as the first argument on the command line. A sample invocation:

```
pgrep '[mM]aynard' file1 file2 file3
```

7. [15] Write a Perl script, called `ileaf`, which will interleave the lines of a file with those of another file writing the output to a third file, i.e. one line from first file, one from second file and so on. If the files are of different length then the extra lines are written at the end. A sample invocation:

```
ileaf filein1 filein2 outfile
```

CS 2413 Test 1 Solutions

1. **[10]**

```
sed '/^[ ][ ]*$/d' < in.data | wc
```

2. **[15]**

```
/^#include/!d
s/^#include//
s/[<">  ]//g
```

This could also have been done by invoking sed with the -n switch and the following script:

```
/^#include/s/[<">  ]//g
s/^#include//p
```

3. **[15]**

```
BEGIN{ time=0; bandw=0; storage=0 }
$1 == "maynard" {
    time += $3 - $2
    bandw += $4 + $5
}
$1 == "*storage* && $2 == "maynard" {
    if ( $3 > storage ) {
      storage = $3
    }
}
END{
    if ( time > 10000 || bandw > 100000000000 || storage > 10000000) {
      print "maynard"
      print "Connect Time =", time
      print "\tBandwidth =", bandw
      print "\tStorage =", storage
    }
}
```

4. **[15]** cmd line: file or files with list of directories printout all .c files

```
@dirs = <>;   # foreach #arg(@ARGV) @dirs = a directory name
chop @dirs;   # chop newline
@cfiles = `find @dirs -name '*.c' -print`;
print @cfiles);
```

an array of

5. [20]

(a) @ARGV == ("-xvf", "a+ b+", "file1", "file2", "file3")
(b) $ARGV[2] == "file1"
(c) $#ARGV == 4
(d) scalar @ARGV == 5
(e)
    i. $c == "file1"
    ii. $e == "~~d~~\n"
    iii. $ARGV == "~~file3~~"
(f)
    i. $c[1] == "dir2\n"
    ii. @c == ("dir1\n","dir2\n","dir3\n","dir4\n","dir5\n","dir6\n")
    iii. $d == "dir7\n"

6. [15]

```
$pat = shift;
@tfiles = grep(-f && -T, @ARGV);
FILE:
foreach $file (@tfiles) {
  open(TFILE,$file) || die "Cannot open $file: $!\n";
  while ( <TFILE> ) {
    if ( /$pat/ ) {
      print $file, "\n";
      next FILE;
    }
  }
}
```

7. [15]

```
open(IN1,$ARGV[0]) || die "Could not open $ARGV[0] for read: $!\n";
open(IN2,$ARGV[1]) || die "Could not open $ARGV[1] for read: $!\n";
open(OUT,">$ARGV[2]") || die "Could not open $ARGV[2] for write: $!\n";
$line1 = <IN1>;
$line2 = <IN2>;
while ( $line1 || $line2 ) {
  print OUT $line1, $line2;
  $line1 = <IN1>;
  $line2 = <IN2>;
}
```