CS 3424 Systems Programming
Fall 2020
PRACTICE MIDTERM EXAM
[ANSWER KEY]

1. Answer true or false for the following statements:

F The set of *extended* regular expressions includes the `$` anchor.

F In Bash, `$@` evaluates to the number of arguments passed into the script

T Given the following bash command, the output printed will be "Yes":
```
COUNT=100; if [[ $COUNT -gt 50 ]]; then echo Yes; else echo No; fi
```

F Given the following bash command, the output printed will be "No":
```
COUNT=100; if [ $COUNT > 50 ]; then echo Yes; else echo No; fi
```

T Given the following bash command, the output printed will be "Yes":
```
COUNT=0; if [[ COUNT > 50 ]]; then echo Yes; else echo No; fi
```

T sed uses basic regular expressions by default, and can use extended regular expressions by passing the `-r` or `-E` options

F Bash will use the `test` utility to evaluate the conditional expression: `[[ -r "$filename" ]]`

F The `grep` utility will search recursively through any specified directories

T In AWK, `$4` contains the value of the fourth field in the current record

F In Bash, an exit status of `1` represents success.

T The extended regular expression `{0,1}` is functionally equivalent to the `?` quantifier

F Given `cmd1 && cmd2`, the second command will run regardless of the outcome of the first command

F The regex pattern matching operator in AWK is "=~"

F In Bash, `$0` contains a string representing the entire command line and all parameters

2. Write a Bash filename pattern (not a regular expression) that will only match the two files indicated in bold and underlined:

```
cs2123p1Driver.c  Makefile    p1abc123.o   p1OutExtra.txt
cs2123p1Driver.h  p1          p1Extra.txt  p1Out.log
cs2123p1Driver.o  p1abc123.c  p1Input.txt  p1Out.txt
```

```
p1[aEI]*.[to]*    -or-   p1[aEI]*.{o,txt}
```

3. Write a bash script that will take one or more directory names as arguments, and count the directories *under* each (that is, the number of subdirectories, subdirectories of subdirectories, and so on; do not count the arguments themselves).
Hint: you may use utilities we've discussed in class to help accomplish this.

```
#!/bin/bash

if [[ $# -eq 0 ]]; then
    echo "Usage: $0 <dir> [dir2 ... dirN]" 1>&2
    exit 1
fi

output=`find "$@" -type d | wc -l`
let count=$((output - $#))

echo "count = $count"
```

4. Write a sed command that will remove all lines longer than 80 characters.

```
/.\{81,\}/d
```

5. Write an AWK script that will perform the specified operations to the following data:
    - Remove the last field from every record
    - Compute and print the total count of all personnel, as well as the average age of the personnel

Example input:

```
Joe DiAlo         24 5023.00   180   890349013 07-19-1994
Christian Amun    44 8825.00   255   860028571 11-29-1974
Kale Martinez     32 5929.01   360   015993535 02-06-1986
John Doe          27 852.60    55    585921492 <no_dob> 00-00-0000
```

Example output:

Joe DiAlo 24 5023.00 180 890349013
Christian Amunpour 44 8825.00 255 860028571
Kale Martinez 32 5929.01 360 015993535
John Doe 17 852.60 55 585921492 <no_dob>
Average age = 29.25

```
1 {
2      $NF = ""
3      agesum += $3
4 }
5
6 END {
7      avgage = agesum / NR
8      print "Average age =", avgage
9 }
```

6. Write a sed command that will delete all leading blank lines at the top of a file.

```
/./,$!d
- alternative solution: -
0,/./{h;d};H;z;x;s/^\n//
```

7. Write a find command that will locate all regular files in your home directory and below that contain the phrase "`version`*xx*" in their name, where each *x* represents a digit. For each file, run the "`ls -la`" command on them.

```
find ~ -name "*version[0-9][0-9]*" -type f -exec ls -la \{\} \;
```

8. Write a sed script that will:

   a. In all lines starting with "*PTY", replace the string "*PTY" with "/*" characters and end the line with "*/"
   b. In the lines starting with "*PTY", replace "X11R6" with "XFree86"
   c. Do not replace "X11R6" on any other lines.

```
/^\*PTY/ {
   s__/\*_
   s_$_\*/_
   s_X11R6_XFree86_g
}
```

9. Given an *excerpt* from a web log (named `part.log`), use a single pipeline to count the number of accesses made from the user with username `root`.

Hint: the command `wc -l` will count the number of lines it reads from standard input.

```
Time                User    Address         Req       Path
12740663209374.890  root    168.29.172.18   HTTP/GET  /www/img/*.gif
12740703137488.990  sam     129.82.45.103   HTTP/GET  /www/index.htm
12740897683420.004  hongyu  66.5.232.10     HTTP/POST /www/root/profile
12741398476528.413  root    129.66.140.111  HTTP/GET  /home/cgi-bin/login.pl
```

```
grep " root " part.log | wc -l
grep -w root part.log | wc -l
awk '$2 == "root" { n++ } END { print n }' part.log
```

10. You run your own mail server on your home machine and you have had a number of sites attempting to relay mail through your server, dryad. So you want a simple sed script that will show the date of each relay attempt as well as the claimed source and destination accounts. ***While there are many different types of logged information in this log file***, the lines relating to a rejected relay attempt is characterized by "Relay denied". These lines are similar to:

```
Feb 26 08:45:59 dryad Relay denied; from=<dlh@email.cq.cnt> to=gogo@ln.com.tw
```

The corresponding output should be:

```
Feb 26 08:45:59 dlh@email.cq.cnt -> gogo@ln.com.tw
```

Write a sed script that will transform each line of input to produce the specified output.

```
/Relay denied/ {
          s/dryad Relay denied; from=<//
          s/> to=/ -> /
}
```

-or-

```
/Relay denied/ s/> to=/ -> /
/Relay denied/ s/dryad Relay denied; from=<//
```

11. Will the following grep command find a match, no match, or produce an error?
    `echo xyz | grep -E ".(dog|cat|bat)*y(horse|pig)?"`

    Match: xyz

12. Write a bash script that takes a filename as a parameter. After verifying that the file exists and is readable, use the "`wc -l`" command in order to obtain a line count for each file. Find the longest file (in terms of their number of lines), and print this file's name, as well as its length in lines. *(Don't worry about breaking ties for files with equal numbers of lines).*

```
#!/bin/bash

if [ $# -lt 1 ]; then
  echo "Usage: $0 <file1> [file2] ... [fileN]" 2>&1
  exit 1
fi

let maxlines=0
for file; do

   # Old-style conditional expression first, new-style second
   #if [ \( ! -r "$file" \) -o \( -d "$file" \) ]; then continue; fi
   if [[ ( ! -r "$file" ) || ( -d "$file" ) ]]; then continue; fi

   length=$(wc -l < "$file")

   if [ $length -gt $maxlines ]; then
       maxlines=$length
       maxfile="$file"
   fi
done

echo "longest file:   $maxfile"
echo "length (lines): $maxlines"
```

13. Write a sed command that will prepend the following line prior to any line that might contain a social-security number. I.e., any line that matches this pattern: `[0-9]{3}-[0-9]{2}-[0-9]{4}`:

```
*** the following line requires redaction ***
```

Example input:

```
…the patient's SSN is 435-32-7992, DOB is 7/12/89, …
```

Example output:

```
*** the following line requires redaction ***
…the patient's SSN is 435-32-7992, DOB is 7/12/89, …
```

```
/[0-9]{3}-[0-9]{2}-[0-9]{4}/ i\
*** line requires redaction ***
```

-or-

```
/[0-9]{3}-[0-9]{2}-[0-9]{4}/ i *** line requires redaction ***
```

-or-

```
/[0-9]{3}-[0-9]{2}-[0-9]{4}/ s/^/*** line requires redaction ***\n/
```

14. Write a sed script that will print the real names of up to the first 100 users whose shell is `bash` (the real name is the fifth colon-delimited field; their shell is specified in the last field).
    Example input file *excerpt* (pretend this file is actually hundreds of lines long):

```
huunguyen:x:1021:1020:Huu Nguyen:/home/huunguyen:/usr/bin/nologin
tianyiliu:x:1022:1022:Tianyi Liu:/home/tianyiliu:/bin/tcsh
quadewarren:x:991:1023:Kyle Landry:/home/quadewarren:/bin/bash
yuyutang:x:1024:1024:Yuyu Tang:/home/yuyutang:/usr/bin/nologin
xuelingzhang:x:725:1025:Xueling Zhang:/home/xuelingzhang:/bin/bash
dalton:x:1026:1026:TJ Bashwami:/home/bashwami:/bin/tcsh
michael:x:980:1027:Michael Thompson:/home/michael:/usr/bin/nologin
...
```

Example output:

```
Huu Nguyen
Tianyi Liu
Kyle Landry
Yuyu Tang
Xueling Zhang
TJ Bashwami
Michael Thompson
…
```

```
1,100 s/.*:.*:.*:.*:(.*):.*:.*/\1/p  # use only with -n
```

-or-

```
1,100 {
    s/.*:(.*):.*:.*$/\1/
    p
}
d
```

-or-

```
s/.*:.*:.*:.*:(.*):.*:.*/\1/
101,$ d
```

Challenge: why does this not work correctly (i.e, as expected)?

```
1,100 {
    s/.*:(.*):.*/\1/
    p
}
d
```

15. Redo the previous problem using AWK instead of sed.

```
1 BEGIN { FS = ":" }
2 1,100 {
3         if($7 == "/bin/bash") {
4             print $5
5 }
```

16.

b. Write a sed script that will print the real names of up to the first 100 users whose shell is `bash` (the real name is the fifth colon-delimited field; their shell is specified in the last field).
Example input file *excerpt* (pretend this file is actually hundreds of lines long):

```
huunguyen:x:1021:1020:Huu Nguyen:/home/huunguyen:/usr/bin/nologin
tianyiliu:x:1022:1022:Tianyi Liu:/home/tianyiliu:/bin/tcsh
quadewarren:x:991:1023:Kyle Landry:/home/quadewarren:/bin/bash
yuyutang:x:1024:1024:Yuyu Tang:/home/yuyutang:/usr/bin/nologin
xuelingzhang:x:725:1025:Xueling Zhang:/home/xuelingzhang:/bin/bash
dalton:x:1026:1026:TJ Bashwami:/home/bashwami:/bin/tcsh
michael:x:980:1027:Michael Thompson:/home/michael:/usr/bin/nologin
...
```

Example output:

```
Kyle Landry
Xueling Zhang
...
```

```
1,100 {
    s/.*:.*:.*:.*:(.*):.*:\/bin\/bash$/\1/p  # use only with -n
}
```

-or-

```
1,100 {
    /:\/bin\/bash$/ s/.*:.*:.*:.*:(.*):.*:.*/\1/p  # use only with -n
}
d
```

-or-

```
1,100 {
    /:\/bin\/bash$/ {
        s/.*:(.*):.*:\/bin\/bash$/\1/
        p
    }
}
d
```

-or-

```
/:\/bin\/bash$/! d
/:\/bin\/bash$/ s/.*:.*:.*:.*:(.*):.*:.*$/\1/
101,$ d
```

17. Find the match for **group 1** in the given string and regular expression. If there is no match, write "NO MATCH." Assume the use of extended regular expressions.

String	Pattern	Group 1
abcd123+456efgh	`/([a-c]+[d-q].)/`	abcd1
abcd123+456efgh	`/[0-9]*([a-d]+)/`	abcd
abcd123+456efgh	`/(^[0-9]*.{3}).*$/`	abc
abcd123+456efgh	`/d.([0-9]*)([1-3])/`	2
abcd123+456efgh	`/([e-g678]+).$/`	6efg