UNIX SYSTEM PROGRAMMING MIDTERM
Spring 2014

The following assumptions can be made throughout the exam:
No variables have been declared other than those shown to be declared in the specific problem.
The shell is always tcsh.
Any scripts shown have already been chmod-ed and are executable.

In answering the output for the various questions:
- If you believe that a specific problem produces an error message, say "ERROR".
- If you believe it produces no output, say "NONE".
- If you believe that there is an empty line, say "EMPTY."
- If you believe that the last line of output does not advance to the next line, say "NONEWLINE."
- If you believe that the output is the same as the input, type "SAME."
- If you believe that the output freezes waiting for user input, type "FREEZE."
- If an answer is on several lines, then put your answer on several lines.

For example:
% echo 1; eecchho 2; echo 4; cat; echo 5
1
ERROR
4
FREEZE
% echo 1; echo ; echo –n 3
1
EMPTY
3NONEWLINE

Some reminders:
The grep -o flag prints only the matching pattern, not the rest of the line.

```
1.
Using the fewest number of keystrokes, write a single command to flip
the upper and lower case letters. For example:
% echo "aBcDeF GH, ijk." | <your command>
AbCdEf gh, IJK.

2.
We want a short script that will count the number of directories in my
path.
For example, if my path is:
/home/stevewhaga/dir1/dir2/dir3
Then here is the desired behavior:
%./dircount
5
```

part a:
Write your script in this format:
```
% cat dircount
#!/bin/tcsh
___ | tr -cd ___ | wc -__
```

Note that you may only fill in the blanks. The first blank is a command, the second blank is a parameter, and the third blank is a flag. Note also that you can answer the 2nd and third blanks, even if you don't remember the first.

part b:
Redo the script from part a, but this time using the following format:
```
% cat dircount
#!/bin/tcsh
___ | tr ___ | wc -__
```

Note that, this time, the tr command cannot use any flags.

3. We want a short script that will strip off everything but the beginning numbers of a line created by grep –n. Note: the file itself is generic; it can contain any characters (even numbers).

For example, Suppose that we have a file with contents such that:
```
% grep -n X Y
2:X
8:XX
205:X
```

Well then, your script will have the following behavior:
```
% grep -n X Y | ./justnumbers
2
8
205
```

part a:
Write your script in this format:
```
% cat justnumbers
#!/bin/tcsh
grep -___  ____
```

Here,there are two blanks, one for flag(s), the other for parameters(s)

part b:
Redo the script from part a, but this time using the following format:
```
% cat justnumbers
#!/bin/tcsh
cut -___  ____
```

Here,there are two blanks, one for flag(s), the other for parameters(s)

part c:
Redo the script again, but this time I tell you that the input file has no numbers in it (other than the numbers at the beginning of each line). This time, use the following format:
```
% cat justnumbers
#!/bin/tcsh
tr -__ ____
```

4. Now we imagine your justnumbers script is placed in your home directory, with executable permission. And we change to a new directory, and then type the following:

```
% ls -l
???  .  ..  *  ac  a.c  b
% cat -n ac
     1  -n
     2  ac
     3  bF\d
     4  F
     5  EBF
     6
     7  j*L
```

Note that this file contains no spaces or tabs, and that line 6 is EMPTY.

What will be the output for each of the following commands:

```
a. tail -n `head -5 ac | wc -w` ac | head -1
b. echo `ls ?`
c. ls .
d. ls ...
e. ls ?
f. ls *
g. ls a
h. ls a*
i. ls a.*c
j. ls `ls ?`
k. echo `head -2 ac`
l. cat `head -2 ac`
m. grep -n a* ac | ~/justnumbers
n. grep -n "a*" ac | ~/justnumbers
o. grep -n "E*F" ac | ~/justnumbers
p. grep -n "F*$" ac | ~/justnumbers
q. grep -n "^F*" ac | ~/justnumbers
r. grep -n "[^F]" ac | ~/justnumbers
s. grep -n . ac | ~/justnumbers
t. grep -n . ac | ~/justnumbers
u. grep -n \\* ac | ~/justnumbers
v. grep -n \\\* ac | ~/justnumbers
w. grep -n \\\\* ac | ~/justnumbers
x. grep -n \\\\\* ac | ~/justnumbers
y. grep -n \\\\\\* ac | ~/justnumbers
z. grep -n \\\\\\\* ac | ~/justnumbers
aa. grep -n \\\\\\\\* ac | ~/justnumbers
bb. grep -n \\\\\\\\\* ac | ~/justnumbers
cc. grep -n \\\\\\\\\\* ac | ~/justnumbers
dd. grep -n '\*' ac | ~/justnumbers
ee. grep -n '\\*' ac | ~/justnumbers
ff. grep -n '\\\*' ac | ~/justnumbers
gg. grep -n "\\\\*" ac | ~/justnumbers
hh. set B = A; set C = B ; grep $C ac
ii. echo \\\\\\
jj. echo a   b""""
kk. echo "can\'t"
ll. echo "He said, "Hi" to me."
mm. wc `ls` | wc
```

```
nn. set B = A ; echo cat ac
oo. set B = A || echo cat ac
pp. set B = A && echo cat ac
qq. @ B = A ; echo ?
rr. @ B = A || echo ?
ss. @ B = A && echo ?
tt. @ B = 0 ; echo ?
uu. @ B = 0 || echo ?
vv. @ B = 0 && echo ?
ww. set T = ( * ); echo $T[-2]
xx. set T = ( 1 2 3 4 ); echo $T[1][3]
yy. set T = ( 1 2 3 4 ); echo $T[0]
zz. echo $#T
aaa. echo $?T | echo $?
bbb. echo 'ab*c' | fgrep -o b\*
ccc. echo 'ab*c' | fgrep -o "b*"
ddd. echo '? ? ? ? ?' | grep -o "? ?"
eee. echo '? ? ? ? ?' | grep "? ?"
fff. echo 'abc' | grep "x*"
```

5. Now we are in a different directory and we type:

```
% ls
0  1  2  3  4
% cat 0
#!/bin/tcsh
echo $#
% cat 1
#!/bin/tcsh
echo '$*'
% cat 2
#!/bin/tcsh
exit $?
% cat 3
#!/bin/tcsh
exit $2
% cat 4
#!/bin/tcsh
echo $<
```

What is the output of the following:

```
a. ls | xargs echo
b. ./*
c. seq `./*`
d. seq 2
e. echo "*" | xargs .\1
f. echo -n "" ; ./0 `echo "*"`
g. echo -n "" ; ./1 `ls` || echo OR
h. echo -n "" ; ./1 `ls` && echo AND
i. echo -n "" ; ./2 `ls` || echo OR
j. echo -n "" ; ./2 `ls` && echo AND
k. echo -n "" ; ./3 `ls` || echo OR
l. echo -n "" ; ./3 `ls` && echo AND
m. ./4 `ls`
n. ./4 < `ls`
o. ./0 < `ls`
p. expr 1+2
```

```
q. ls | head -1 | xargs expr 1 * 2
r. ls | head -1 | xargs expr 1 *
s. ls | head -2 | tail -1 | xargs expr `ls | tail -1` +
t. expr `expr `expr 1 + 2` + 3` + 4
u. ls | tee echo > f ; ls > g ; diff -y f g
```

For this last one (u), note that the syntax of diff -y is to display the 2 files side-by-side, according to the following syntax:

```
% cat f1
b
x
d
f
g
% cat f2
b
c
d
e
f
g
% diff -y f1 f2
b                                                                 b
x                                                             | c
d                                                                 d
                                                              > e
f                                                                 f
g                                                                 g
```