SYSTEM PROGRAMMING MIDTERM
Spring 2009

1. (12 points total)

a What is the output of the following:

```
% cat echoes
#!/usr/bin/tcsh
echo "s/['"'"\!$%&(){}\\\/?.,<>;:~]/|&/g'        (4pts)
echo $1                                          (1 pt)
echo \$1                                         (1 pt)
echo "$1"                                        (1 pt)
echo '$1'                                        (1 pt)
echo $1    $3$2                                  (1 pt)

% ./echoes cat dog bird
```

b. (1 pt) Give the echo command to print:

```
He said, "Hi!"
```

c. (2 pts) Give the echo command to print:

```
He said, "she said, 'Hi!'"
```

2. (10 points total) Write grep commands to find each of the following:

a. (1 pt) empty lines

b. (2 pts) lines that contain at least one integer, as a whole word.

c. (3 pts) an "a" followed later (but not immediately) by a "b".
So these lines match: "abad ball" and "a7b" match
But these lines do NOT: "baby", "abe b", "apple", "bay"

d. (2 pts) lines that contain one of the following subsequences: "BAKE", "BIKE", "BATE", or "BITE". But these are subsequences not substrings. So all of the following are matches:

This line B mATchEs
So does this BI line TTE.
And so does this one BIKE
BAnd this KE one

e. (2 pt) lines that do not end with either "stop" or "Stop"
Note: lines ending in "STOP" would print
Note: the quotation marks are not part of the string.

3. (3 pts) Write an egrep expression to find either an "a" followed (not immediately) by a "b", or a "b" followed (not immediately) by an "a".

4. (3pts) Write the ls command to list all files that have an "a" followed later by a "b". Unlike 2c and 3 however, you are allowed to have a "b" immediately after the "a". But that "b" will not count in the match.

So these lines match: "abad ball" , "a7b", abe b" match
But these lines do NOT: "baby", "apple", "bay"

5. (6 pts total) Write the following sed commands:
a. (2pts) To remove the second letter on each line
b. (2pts) To put "[" and "]" around the second letter on each line
c. (2pts) To print only lines containing a "//"-style comments

6. (14 pts total) Explain what does each of the following do (f & g are in lex):
a. (2pts) `awk '/Beth/{n++}; END {print n+0}' file`
b. (2pts) `awk 'BEGIN{ORS="\n\n"};1'`
c. (2pts) `awk 'NF > 4'`
d. (2pts) `awk '{temp = $1; $1 = $2; $2 = temp}' file`
e. (2pts) `awk '!($0 in a) {a[$0];print}'`
f. (2pts)
```
%%
[ \t]+$ ;
[ \t]+ printf(" ";
```
g. (2pts)
```
%%
[a-zA-Z][a-zA-Z0-9]* printf("got one");
```

7. (2 pts) Give an example of an expression that can be described with yacc but not lex.
Give an example of an expression that cannot be described with yacc.

8. (12 points total) Suppose that you are modifying a large C++ program. The source directory contains .cpp files, .h files, a Change log file and various other file types.

a. (1 pt) In the process of your work, you want to find every place where the variable VarNum occurs in any of the files. Write the Unix command to find all such lines, along with their line numbers, assuming you are in this directory.

Your output might look something like:
file1.c:24: // VarNum should be 3
file1.c:25: if (VarNum!=3)
file1.h:124: int VarNum
file2.cpp:32: /* This method does not work for all VarNum
file2.cpp:33: sizes. It works for typical VarNum values
file2.cpp:34: but it will not work for VarNum > 7 */
file7.def:213: VarNum = 7; // Note this change should be added to the ChangeLog
ChangeLog:6:Also added a check for VarNum == 3

Note: this is just a sample output for illustration.

b. (2 pt) The above output lists a line that is in the ChangeLog. The ChangeLog file is not really part of the program code, so maybe you don't want to see this one. Assuming that part a is correct, pipe the answer into a command to remove lines from matches to the ChangeLog file.

Note: the sample output provided in part a lists the word ChangeLog twice, but only one of these should be suppressed, because the other is in file7.def.

c. (3 pt) Suppose that you want to only know the names of files that match. Assuming that parts a and b are correct, pipe its result into Unix command(s) to just list the names of the files that match. Print each filename once.

Note: You cannot use grep –c, because we did not teach it.
Note: The output from part a has a colon after each file name

d. (1pt) Pipe the result of part c into a Unix command to display all of the contents of all of the files in the list, along with line numbers.

e. (5 pts) You now have a list of files that contain the word VarNum. But some of the matches are within comments. Suppose you want to see only real uses of the VarNum, not comments. Assume that the output of part d is piped into awk –f prog.awk.
Write the prog.awk file, so that only lines that really use VarNum are printed.

In our example, the output might look something like:
file1.c:25: if (VarNum!=3)
file1.h:124: int VarNum
file7.def:213: VarNum = 7; // Note this change should be added to the ChangeLog

9. (10 pts) A user types: `ls -l | awk -f program.awk`
Write the awk program so that the files are listed according to extension. Files with no extension should list first.

As a hint, recall that this is the format of ls:

% ls -lrt
total 73949
-rw-r--r-- 1 Administrator None 131 Dec 18 2008 ExtensionIsLl.ll
-rw-r--r-- 1 Administrator None 131 Mar 18 02:49 ExtensionIsC.c
-rw-r--r-- 1 Administrator None 735 Apr 4 20:31 ThisHasNone
…
Note: You will want to use an array based on the extension, and another array to list what the extensions are.

10. (8 points) Write a tcsh script to ask for a file name. If the file does not exist, print an error and exit. If it does exist, run the file. If the run was not successful, print an error.

11. (20 pts total) Mostly from the homework. Describe what does each of the following does in one or two sentences. (Not what it did in the original homework, but what each of these lines does by itself.)

The following are from tcsh shell scripts:

a. (1 pt) `if ($word == $argv[$#argv]) then`

b. (2 pts) `ls -l ./*/*/*.c | sort -nk5`

c. (1 pts) `tr -c "a-zA-Z" "\n"`

d. (2 pts) `sed '2,$ s/.*/&/p'`

e. (2 pts) `sed -e 's/\([0-9]*\):.*/\1/'`

f. (2 pts) `sed 's/^[A-Z]/| &/'`

g. (2 pts) `sed -e 'N; s/\n[0-9]*\:/|/'`

h. (1 pts) `echo '"(a|b)" ht' | xargs egrep`

The following is an awk pattern/action pair:

i. (2pts) `NR==1 {if(index($1,"G")==1) fromgerm=1}`

The following are from inside the action parts of awk files:

j. (2pts) `if (!(substr(german,2) in word))`

k. (1pts) `$i = word[$i]`

l. (1pts) `german = german " " $i`

m.(1pts) `if ($j ~ "::")`