-20

Anirshi #1

Name Karl Estes ID 1167854

CptS 224
Final Exam
Tuesday, December 11, 2018 3:10-5:00pm

143
163

There are 27 questions on 9 pages **(please check NOW to see that you have all pages).** Each question is worth 3 points unless otherwise noted. 163 points (a prime number) total (not just a prime, but a strong prime, a lucky prime, and a fortunate number).

If you need extra space, write on the back side of a page. But make a note on the space below the question (i.e., where you ran out of room) "pointing" the grader to the other page.

This test is CLOSED book, CLOSED neighbor, CLOSED laptop/cell/etc

Note: don't freak out, this class's grades WILL be curved…

✓ 1. What does $0 represent in Bourne Shell?
The Script that is being run

✓ 2. What does $0 represent in awk?
The entire line that was read in

✓ 3. What does $1 represent in awk?
The first field of the line read in

✓ 4. What does $1 represent in Bourne Shell?
The first argument that was passed in

? -1 5. How do you spell "else if" in Bourne Shell?
elseif elif

3 6. What does $3 represent in Perl?
The 3rd argument that was passed in to the command

3rd match

-1 7. What does $2 represent in awk?
The 2nd field of the line that was passed in

X

Read in - passed in Sands like parameter

✓ 8. How do you spell "else if" in Gaelic? **[0 points**, but impressive nevertheless]

✓ 9. How do you spell "good luck" in ~~Perl~~Persian? **[0 points**, but impressive nevertheless]

H

Given the following file `the_ant` (the "<>" is not really part of the file contents, it is just to denote a blank line):

<>
The Ant
by Ogden Nash
<>
<>
The ant has made himself illustrious
Through constant industry industrious.
So what?
Would you be calm and placid
If you were full of formic acid?
<>

10. **[9 points]** What would be the output of the following command?

```
$ awk '
BEGIN {
    count=0;
    total=0;
  }
  { total++; }
/[aeiou][aeiou][aeiou]/ {
    print "Found a match! ->", $0;
    count++;
  }
  END {
    print "I counted", count, " lines of", total, " that match."
  }'  the_ant
```

Found a match -> The ant has made himself illustrious
Found a match -> Through constant industry industrious
I counted 2 lines of 11 that match.

11. **[6 points]** What would be the output of the following command?

```
gate1 % tail -4 the_ant | sed -e 's/acid/kool-aid/g'
```

ious
ious
kool-aid
kool-aid

12. **[12 points]** Give 4 ways in Perl the following key/value pairs representing test scores into an associative array (you may give it any name). (You don't need anywhere near a full page, room is given nevertheless.)

Key	Value
JoeVandal	50
ButchTheCoug	100
WhitepawTheMalmute	0

```
$mascot{JoeVandal} = 50;
$mascot{ButchTheCoug} = 100;
$mascot{WhitepawTheMalmute} = 0;

%mascot = (JoeVandal, 50, ButchTheCoug, 100, WhitepawTheMalmute, 0);

%mascot = (
    JoeVandal       => 50
    ButchTheCoug    => 100
    WhitepawTheMalmute => 0
);

@mascot(JoeVandal, ButchTheCoug, WhitepawTheMalmute) = (50, 100, 0);
```

13. **[10 points]** Which do you believe is more powerful: a GUI interface (Windows, Mac, Linux GUI) or a command-line interface such as with Linux?

A command-line interface such as with Linux is more powerful than a GUI interface. A GUI interface limits you by what options your given in the GUI. It's easy to open applications and navigate, especially for a non-tech person, but a command line interface gives you more control over your actions and what the computer does. It's easier to breakdown files (search them for instances of certain characters and sort accordingly) and to change file permissions as two examples.

14. **[12 points]** You have learned the Bourne shell script language, awk, sed, grep. Many tasks can be done by most or all of these, though some have features others don't, and some are more suitable or terse or somehow preferable over the others in some cases.

In 4-6 concise, well-considered sentences, outline a particular task that you would definitely prefer to use one of these languages to do instead of the others, and tell why. There is of course more than one answer that can receive full credit.

While all the languages listed above are powerful, if I was given the task of editing a file and removing or replacing certain names or words, I would choose sed. Sed is designed as an editor and makes it simple to exchange a word for another. By knowing the exact word or regular expression pattern I am looking for, I need simply supply the alternate (or none for a delete) and set the parameter to global. While this can be done in the other languages, it would take a series of statements while sed can perform the same task with one line of code

---

15. What is a list in perl, and what is it used for?

A list in perl is a collection of scalar values seperated by commas. A list can be used to group like items and can be used to initialize arrays and associative arrays.

16. What character must the names of the following perl "types" begin with?

Scalar: $

Array: @

Associative Array: %

Given the following shell script (stored in file `answer`):

```
#!/bin/sh

case $1 in
[Yy]es|[Ss]i|[Jj]a)
        answer="Yes"
        ;;
*)
        answer="No"
        ;;
esac

if [ "$answer" == "Yes" ]
then
        echo "Thank you for agreeing with me!"
else
        echo "Are you sure I can't change your mind?"
fi
```

17. What would be the output of the following command?

```
./answer Si
```

✓ Thank you for agreeing with me

18. What would be the output of the following command?

```
./answer yes never
```

✓ Thank you for agreeing with me

19. What would be the output of the following command?

```
./answer Y
```

-3 Thank you for agreeing with me

20. What would be the output of the following command?

```
./answer sure
```

✓ Are you sure I can't change your mind

-3

21. **[15 points]** Write an `awk` script that will print out the word, line number, and filename for any duplicate words which exist next to each other. It does not have to deal with punctuation (note "very" is not in the sample output below): you can use the standard awk whitespace field delimeters

For example:

-3

input.txt file:

…

line 13: The sea was was very, very blue blue today.

…

NR = # of records } Don't remember if these are right but
NF = # of fields } this is what I'm going with

your output:
was 13 input.txt
blue 13 input.txt

```
$ awk '
    BEGIN {
        line=0; File=FILENAME;
    }
{ line++;
  field = $1;
  Count = 1;
}
while (count -le NF)
{
    if (field == ($field+1))
    {
        Print $field, " ", Count, " ", File;
        Count++;
        $field++;
}   FILENAME
```

-3

22. **[30 points]** For the following English patterns:

1. *Start of line/filename, then any number of lower case letters, one decimal digit, then end of line/filename.*
2. *The letter 'c', followed by one of any single legal character, followed by a decimal digit, followed by zero or more of any legal character, then a 'Z'.*
3. *A blank line (regex) or a zero-length filename (shell pattern).*
4. *Start of line/filename, then 'H', then any number of legal characters, then 'd', then end of line/filename.*
5. *'T' must be somewhere, but not at the start of the line/filename.*

Give both a regular expression and a shell file pattern (e.g., to use with `ls` or `wc`) that matches it. If it is not possible to specify the pattern in that "language", then draw a line through that box. Also, if Unix does not allow a filename to be what is described above, write "ILLEGAL" in the box.

Note: *legal character* means any symbol that is legal to be in a filename or in a file; you should not have to enumerate them and will get a point off if you do – there is one symbol you can denote this with). Also, for that matter, you should not have to "brute force" enumerate anything.

Hints:

- There are 2 of these that can't be expressed with a shell file pattern and 1 that are an illegal file name. (You're welcome for the gift here: Merry Christmas, Kool Kwanza, Ramadan Mubarak, Happy Hannukah, or "have a great day", whatever applies.).
- You don't need to do anything to specify the end of a filename in a shell pattern, the last part of your pattern matches the end of the filename --- that's how shell file patterns work!

	Regular Expression [4 pts]	**Shell File Pattern [2 pts]**
#1	^[a-z]*[0-9]$	
#2	C.[0-9].*Z	C?[0..9]*z
#3	^$	ILLEGAL
#4	^H.*d$	
#5	.+T	?*T*?

-5

23. **[6 pts]** Describe in a sentence or two each three capabilities that Source Code Management (SCM) software can provide for large software projects that manual file copy cannot.

Unlike with a manual copy, source code management allows every to work on a version of the project at the same time. It protects the main code files from to much editing by having each person clone the project. Lastly, it allows for conflicting changes to be noted before they are merged back into the project.

Given the following Perl script, named 'perlexample.pl':

```
#!/usr/bin/perl

$a = 0;
$b = 1;

print "0 1";

for ($cnt=2; $cnt <= $ARGV[0]; $cnt++) {
   $next = $a + $b;
   print " $next";
   $a = $b;
   $b = $next;
}

print "\n"
```

24. **[6 points]** What will be the output of the following command? [4 points]

```
./perlexample.pl 5
```

0 1 1 2 3 5

-0

Given a file called `the_ant`, with the following contents:

```
The Ant
  by Ogden Nash

The ant has made himself illustrious
Through constant industry industrious.
So what?
Would you be calm and placid
If you were full of formic acid?
```

25. **[6 pts]** What would be the output of the following command?

```
grep '^Th' the_ant
```

The ant has made himself illustrious
Through constant industry industrious.

-1

26. **[6 pts]** What would be the output of the following command?

```
grep '^Th[^r]' the_ant
```

The ant has made himself illustrious

-0 ~same mistake as #25 (left out "The Ant")

27. **[6 pts]** What would be the output of the following command?

```
tr '[M-Z]' '[m-z]' <the_ant
```

the ant
by ogden nash
the ant has made himself illustrious
through constant industy industrious.
so what?
would you be calm and placid
if you were full of formic acid?

I -1

-2