Basic Unix for Engineers
Department Computer Science and Engineering
University of Notre Dame

due March 6^{th}, 2015 (by class time)

MIDTERM

THIS EXAM SHOULD BE AN INDIVIDUAL EFFORT. You should NOT discuss answers with your classmates, nor post questions in piazza. All questions should be directed to the teaching staff. Similar answers will be considered academic honor code violations.

You may consult the lecture notes, the textbook, posted homework solutions, and man pages.

For submission, include the following files in the `midterm` directory in your dropbox:

- `midterm.tex` with your answers. Typeset you submission in LaTeX according to Problem 1.
- `midterm.pdf`, obtained by compiling `midterm.tex`.
- All of the scripts requested as individual files.

You can find sample input and output files at:
`dropbox/NETID/midterm/0[3,4,5,6]`

Problem 1. [10 points] The answers to your midterm should be typeset in LaTeX. Your document should compile correctly with `pdflatex`, using a section per problem. Use subsections or lists for problems' sub-parts.

Scripts and command lines should be listed using the `listings`. Use the `language` option as it is appropriate per problem.

Problem 2. Short questions.

(a) [1 points] How do you exit from `vi` without saving any changes?

(b) [1 points] In `tcsh`, how can you re-execute the last command line without typing it again?

(c) [2 points] What is the purpose of the PATH variable in a Unix shell?

(d) [2 points] Give two differences between shell and environment variables.

(e) [2 points] Give two differences between basic and extended regular expressions.

(f) [2 points] Identify a possible issue in the command line: `grep [0-9][0-9]* myfile.txt`

(g) [5 points] What are the differences between single, double, and backtick quotes in `tcsh`?

Problem 3. [10 points] Give a `sed` command line that converts dates from the format `MM/DD/YYYY` to the format `YYYY/MM/DD`. You may assume that all of the dates are valid (e.g., you do not need to check whether a month is between 01 and 12).

Dates should follow the format `MM/DD/YYYY`, and appear as 'words' by themselves. Dates that do not strictly follow this format should not be changed. For example:

Sample input:

```
04/24/1324garden 05/14/1943.
08/16/1471
10/xy/1288 02/14/1407
resonance 11/20/1285
01/15/1360 01/15/360 01/5/1360 1/05/1360 11/20/1498
01/13/126308/15/176006/26/1270
caixinha10/13/1828 tuscan upbuild
05/25/1329 houses/09/1285
09/ab/2007 tables apples
04/26/word
```

Sample output:

```
04/24/1324garden 1943/05/14.
1471/08/16
10/xy/1288 1407/02/14
resonance 1285/11/20
1360/01/15 01/15/360 01/5/1360 1/05/1360 1498/11/20
01/13/126308/15/176006/26/1270
caixinha10/13/1828 tuscan upbuild
1329/05/25 houses/09/1285
09/ab/2007 tables apples
04/26/word
```

Partial credits:

5 Identify dates' components.

5 Switch positions of month, day, and year.

Problem 4. [11 points] Write a `tcsh` script named `file-sum` that takes any number of filenames as arguments and echoes the sum of their sizes, in bytes.

As soon as the script finds that a file does not exist, it is not a regular file, or it is not readable (in that order), the script should print an error message that describes the type of error and return immediately with status 1.

It is not an error if the script is called without any arguments. In such case the script should print 0.

If no error is found, the script should exit with status 0.

Partial credits:

3 Identification of files to be added up.

3 File sizes inquiry.

3 Arithmetic.

2 Return statuses.

Problem 5. We say that a directory is *marked* if it contains a file called `.mark` or if it has an ancestor that contains such a file. For example:

```
/
|-A
|---B
|   |-.mark
|-C
|-D
  |-E
  | |-.mark
  | |-F
  |   |-G
  |     |-H
  |     |-I
  |     |-J
  |-K
```

A, C, D and K are not marked. B and E are marked since they contain a file called `.mark`. F, G, H, I, and J are marked via E.

(a) [17 points] Write a `tcsh` script named *look-mark* that receives a directory absolute path as an argument, and determines whether such directory is marked. The script should exit with status 0 if the directory is marked, status 1 if it is not marked, and status 2 if the argument missing, it does not exist, or it is not a directory.

For example:

```
$ ./look-mark /D/E/F/G/H
(returns status 0)
$ ./look-mark /A
(returns status 1)
$ ./look-mark /D/X/Y/Z
(returns status 2)
```

(b) [2 points] The `readlink` command with the `-f` option prints the absolute path of a given relative path. Based on *look-mark*, create the script *look-mark-rel* that works even if its argument is a relative path.

Partial credits:

17 Mark finding.

- **2** Process number of required arguments.
- **3** Test the existance of a file.
- **3** Find path components.
- **7** Logic and flow control to return the desired return status.

2 Correct use of `readlink -f`.

Problem 6. [35 points] Trolls in the Discworld have a number system based on the number 4. The numerals are:

Numeral	Decimal value
`one`	1
`two`	2
`three`	3
`many`	4
`lots`	16

Numerals are separated by '-', they appear in descending order, and their values are simply added together. For example, for 10, trolls would write `many-many-two`, and for 75, they would write: `lots-lots-lots-lots-many-many-three.`

Write an `awk` script name `troll-num` that verifies the decimal values of a list of troll numbers. For example, for input:

```
one : 1
mAny-two : 6
lOts-lots-lots-LOTS-lots-loTs-many : 101
tWo : 3
```

Your script should print to standard output:

```
one : 1 : correct
mAny-two : 6 : correct
lOts-lots-lots-LOTS-lots-loTs-many : 101 : incorrect
tWo : 3 : incorrect
```

Columns are separated by ' : ' (a colon surrounded by single spaces):

- The first column has the troll numbers. Numerals may appear in upper or lower case (e.g., `two`, `Two` and `tWO` are equivalent).
- The second column has the decimal values to test.
- Your script should add a third column, indicating whether the troll and decimal numbers are equivalent.

Your answer should include a function that receives a string in troll numerals and returns its decimal value.

Partial credits:

5 Reading the input with the correct column separators.

5 Writing the output with the correct column separators.

20 Function that correctly converts troll numbers to decimal values.

- **5** Correctly writing and calling a function.
- **5** Correctly splitting a troll number into its components.
- **10** Correctly converting a troll number into its decimal value.

5 Correct logic of the correct/incorrect column.