

CSC 209 Assignment 1, Fall 2011: Shell scripts

Due by the end of Friday October 14, 2011; no late assignments without written explanation.

Please re-read the statement about academic offences on the course information sheet!

Part one: Zeller's congruence

Write a shell script which implements "Zeller's congruence" and tells the user the day of the week corresponding to the argument date.

Example:

```
% sh zeller 2011 10 14
Friday
%
```

Zeller's congruence is the following algorithm:

1) First, adjust the date so that the beginning of the year is March, called month zero. For example, March 2011 would have a year value of 2011 and a month value of 0; February 2011 would have a year value of 2010 and a month value of 11; October 2011 would have a year value of 2011 and a month value of 7.

2) Calling the adjusted values "newyear" and "newmonth", and given a one-origin (i.e. normal) "daynumber" value, compute:

$$\begin{aligned} & ((26 * (\text{newmonth} + 1) - 2) / 10 + \text{daynumber} + (\text{newyear} \% 100) \\ & + (\text{newyear} \% 100) / 4 + \text{newyear} / 400 - 2 * (\text{newyear} / 100) + 7777) \% 7 \end{aligned}$$

where '/' indicates integer division.

Or, more nicely formatted:

$$\begin{aligned} & (26 * (\text{newmonth} + 1) - 2) / 10) \\ & + \text{daynumber} \\ & + (\text{newyear} \% 100) \\ & + (\text{newyear} \% 100) / 4 \\ & + \text{newyear} / 400 \\ & - 2 * (\text{newyear} / 100) \\ & + 7777 \end{aligned}$$

all mod 7. (An example implementation in Python is available on the Q&A web page for definiteness.)

The result is zero for Sunday, one for Monday, etc.

You will find the "mock associative arrays" technique discussed in tutorial to be useful for converting the number to the name of the day.

Your shell script must run under all versions of *sh* and thus may not use *bash*'s built-in arithmetic functions. For simple formulas you can use *expr*; for more complicated expressions (such as the main expression above) I suggest *dc*, as demonstrated in lecture.

You can assume that all command-line arguments are well-formed positive integers, but you do need to check the argument count ($\$ \#$).

(over)

Part two: A perfect game of Nim

Write a shell script to output the next move in the game “Nim” (as discussed in class, and with further discussion under the Q&A web page). The command-line arguments are the numbers of sticks in each of the piles.

Example session:

```
% sh nimmove 10 13 15
I take 8 sticks from pile 3
%
```

To continue to play the game, suppose the user decides to take one stick from the first pile. They would then execute the command

```
% sh nimmove 9 13 7
```

This could, of course, be encapsulated in a larger interactive shell script. For this assignment you will write only the computer-move portion. The algorithm to play perfectly (which you must use) is described under the Q&A web page (and was discussed in class).

To compute an XOR (as required for the strategy), first convert the numbers to binary using *dc*, as in the following example:

```
% echo 13 2o p | dc
1101
%
```

You can then perform an XOR by adding them (with *expr*) as if they were decimal numbers, then using *tr* to change all 2s to 0s.

You can assume that all command-line arguments are well-formed non-negative integers, and that at least one is positive. However, you should emit a usage message in the case of no arguments.

Other notes

For now, we will run our shell scripts by typing “sh file” or “sh file args ...”.

Shell scripts should begin with an assignment to the *PATH* variable as discussed in class. But we will not be worrying about the “#!/bin/sh” thing for assignment one.

Do not use *awk*, *perl*, *python*, or any other programming language in your scripts besides *sh*. (These are valuable languages to know, but are not the point of the current assignment, in which you will be graded on your *sh* programming.)

Please see the assignment Q&A web page at

```
http://www.dgp.toronto.edu/~ajr/209/a1/qna.html
```

for this and other reminders, and answers to common questions.

Remember:

This assignment is due at the end of Friday, October 14, by midnight. Late assignments are not ordinarily accepted and *always* require a written explanation. If you are not finished your assignment by the submission deadline, you should just submit what you have, for partial marks.

Despite the above, I’d like to be clear that if there *is* a legitimate reason for lateness, please do submit your assignment late and send me that written explanation.

I’d also like to point out that even a zero out of 10% is far better than cheating and suffering an academic penalty. Don’t cheat even if you’re under pressure. Whatever the penalty eventually applied for cheating, it will be worse than merely a zero on the assignment.