

Due: Wednesday, 13 February 2019 at 2400

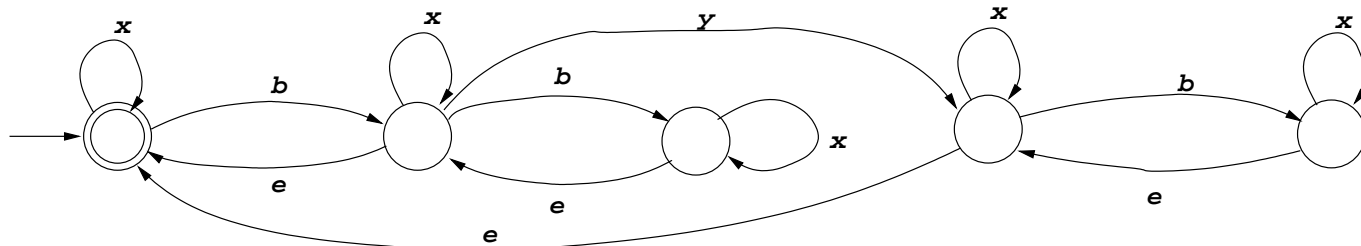
Please submit this homework electronically. See Submitting Your Work in Using Git in CS164 on the class web page.

You'll find templates for some solutions in the directory `~cs164/hw/hw2` and in the **shared** repository. Place answers to any questions that don't require programs in a file called **hw2.txt**.

Problems 1 and 2 call for the use of the **fsasim** program, which is a Python3 program that is available from your instructional account (you can download it from `~cs164/bin`; at home, you'll have to change the first line to wherever you have Python installed, or run **fsasim** using the `python` (or `python3`) command). There is a manual for **fsasim** available from the CS164 home page. The program is also in the **software** branch of the **shared** repository. To get it from there, we suggest you set up a separate directory (let's say `cs164-software` as an example), and use the command

```
git clone -b software cs164-taa@ashby.cs.berkeley.edu:shared cs164-software
```

1. Find the simplest NFA you can for problem 2c in HW#1. Turn in a file called **2c.nfa** containing your answer in **fsasim** format.
2. Find the simplest DFA you can for problem 2c in HW#1. Turn in a file called **2c.dfa** containing your solution in **fsasim** form (see above).
3. Construct a DFA that recognizes all numerals in base 5 (containing only digits 0–4) that are evenly divisible by 4. Turn in a file called **3.dfa** containing your solution in **fsasim** form.
4. Suppose that I have two NFAs, M_1 and M_2 , recognizing the languages L_1 and L_2 , respectively. Give a general algorithm for constructing an NFA that recognizes the language $L_1 \cap L_2$, which is the set of all strings in both L_1 and L_2 . You don't need to write a program, just the give the algorithm in sufficient detail that a reasonable programmer could fill in the details.
5. Give the simplest description you can of the language described by the DFA below:



6. Lecture #4 describes the *pumping lemma*.

- Given a machine, R , what is a suitable value for p (the length at or beyond which a string can be “pumped”)?
- Prove the pumping lemma constructively: that is, describe a procedure that, for any string S with $|S| \geq p$ (with p as determined in part a), finds suitable strings x , y , and z .
- Use the pumping lemma to demonstrate that the set of all strings of the form $a^n b^n$ (a sequence of a ’s followed by the same number of b ’s) is *not* recognizable by a FSA.

7. The configuration file I use for `gitolite`, the system that maintains all our GIT repositories, contains entries that look like this:

```
repo users/cs164-xxx
  RW+      = @admins
  R        = @instructors cs164-raa
  RW refs/tags/ = cs164-xxx
  - refs/tags/  = cs164-xxx
  RW+      = cs164-xxx
repo teams/Team42
  R        = @instructors
  RW+      = cs164-ra @admins cs164-raa
  RW refs/tags/ = @Team42
  - refs/tags/  = @Team42
  RW+      = @Team42
repo shared
  RW+      = @instructors
  R        = @all
```

(The syntax is simplified for the purposes of this problem; in particular, blank lines are not allowed.) The file contains blocks, each starting with a `repo` line designating a repository, followed by one or more access declarations, indicating who may access what parts repository and what they may do to those parts (read, write, modify, etc.). Each clause (`repo` or access declaration) is on one line; blanks may be inserted freely; and blanks separate adjacent words. All names (of individuals and paths) can contain alphanumerics, slashes, periods, underscores, and dashes, and start with an alphanumeric. Group names (`@` followed by a name) may appear only after `=` signs in access declarations. There must be at least one entry to the right of each `=`. To the left of each `=` is one of the strings `R`, `RW`, `RW+`, or `-`, followed optionally by a name (of a GIT reference, if you’re curious).

Write a program using full Python regular expressions to search such a file for cases where there are (at least) two blocks with the same repository name (in their `repo` lines) and remove all but the last of these blocks, using the template file in `cs164/hw/hw1/cleanup.py`. You will probably want to consult the entry in the Python library reference documentation for the `re` module. To (considerably) simplify your task, it is *not* necessary that your pattern rejects syntactically invalid configurations, just as long as it works on correct ones.