1. (*9 points*) For each of the following languages, draw a transition diagram for a DFA that accepts the language. That is, it accepts all the strings in the language and no other strings. (The alphabet for the DFA is the same as the alphabet for the language.)

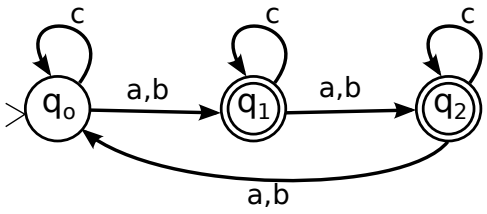
a) $\{w \in \{a,b,c\}^* \mid$ the number of a's plus the number of b's in w is **not** a multiple of 3 $\}$ [Note that the alphabet also includes c!]

b) $\{x \in \{a,b,c\}^* \mid x$ contains a c and there are no a's before the first $c\}$

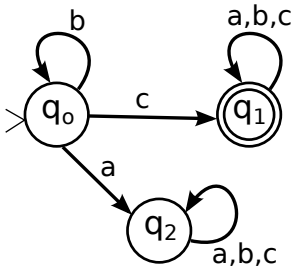
c) $\{y \in \{a,b\}^* \mid y$ ends with the string ababb$\}$

Answer:

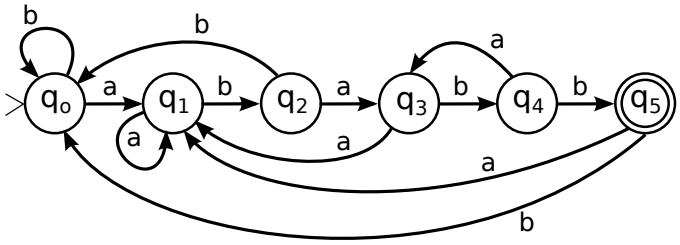
a)



b)



c)



2. (*5 points*) Suppose that a DFA M is defined as $M = (Q, \Sigma, p_1, \delta, F)$, where:

$Q = \{p_1, p_2, p_3, p_4\}$ $\quad \Sigma = \{a, b, c\}$ $\quad F = \{p_2, p_4\}$

and δ is given by the table shown at the right.

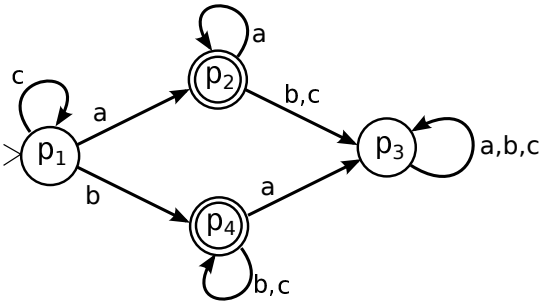
	p_1	p_2	p_3	p_4
a	p_2	p_2	p_3	p_3
b	p_4	p_3	p_3	p_4
c	p_1	p_3	p_3	p_4

a) Draw a transition diagram for M.

b) Based on your diagram, write a regular expression for the language that is accepted by M, and briefly explain your reasoning.

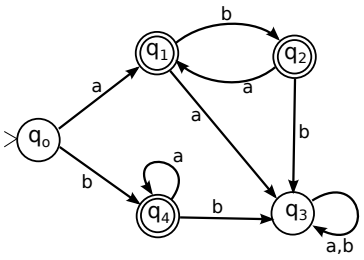
Answer:

a)



b) The language accepted by this DFA is generated by $c^*aa^* \mid c^*b(b|c)^*$ or equivalently by $c^*\big(aa^* \mid b(b|c)^*\big)$. The machine can start by reading any number of c's in state p_1. But to get to one of the accepting states, it must read either an a or a b at some point. If it reads an a, it moves to state p_2, where it can read any number of additional a's, but if it ever reads a b or c, it goes to the junk state p_3, and the string is rejected. If it reads a b in state p_1, it goes to state p_4, where it can read any number of b's and c's, but if it reads an a it goes to the junk state and rejects the string.

3. (*5 points*) Consider the DFA that is defined by the transition diagram shown at the right.



a) Suppose that this DFA is given formally as $M = \{Q, \Sigma, q_o, \delta, F\}$. Identify Q, Σ, δ, and F. For δ, give the transition table.

b) Find a regular expression for the language that is accepted by this DFA. Explain your reasoning.

Answer:

a) $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{a, b\}$, $F = \{q_1, q_2, q_4\}$, and for δ:

	q_0	q_1	q_2	q_3	q_4
a	q_1	q_3	q_1	q_3	q_4
b	q_4	q_2	q_3	q_3	q_3

b) The language accepted by this DFA is generated by $a(ba)^* \mid ab(ab)^* \mid ba^*$. We need to look at how to get from the start state to each of the accepting states. If the machine reads an a in state q_0, it goes to the accepting state q_1, and it can then return to that state by reading ba any number of times. So strings that match $a(ba)^*$ will take the machine to state q_1 and be accepted; furthermore, any string that takes the machine to state q_1 is of this form. Simlarly, the machine can get from q_0 to the accepting state q_2 by reading ab, and it can then return to that state by reading ab any number of times. This gives the regular expression $ab(ab)^*$. Finally, if the first character of the string is b, then the machine moves from state q_0 to the accepting state q_4, where it can stay in that state by reading any number or a's.

The remaining exercises are based on a file, *data.txt*, which you can get from the directory /classes/cs229 on math.hws.edu or through the following link

http://math.hws.edu/eck/cs229/s21/data.txt

This is based on an anonymized file of grade data from the registrar, but the grades have been randomized to make the data totally meaningless. I urge you to actually do the exercises on a computer, but you are only asked to report how the exercises can be done. The first three exercises could be done in a text editor that supports regular expressions. You could, for example, add the file to a project in Eclipse and use Eclipse's search-and-replace feature. You could do the same exercises on the command line in Linux or, I believe, on a Mac, with a command that uses *perl -pe*. The last two exercises require the command line.

4. (*2 points*) Dates in the file are given in the format Month/Day/Year. You would like them to be in the form Day-Month-Year. For example, you would transform 3/17/2021 to 17-3-2021. How could you use regular expression search-and-replace to make the change? What search expression and replacement text would you use?

Answer:

We need to match the month, day, and year separately, so that they can be copied into the replacement. That means that the parts of the regular expression that match them must be in parenthesized groups. Each group is made of one or more digits so can be matched by an expression of the form `[0-9]+`. We also need to match the /'s in the dates. Then the replacement text `\2-\1-\3` will put the groups into the desired order, separated by hyphens.

Search expression: `([0-9]+)/([0-9]+)/([0-9]+)`

Replacement text: `\2-\1-\3`

On the command line, we could do this using

```
cat data.txt | perl -pe 's/([0-9]+)\/([0-9]+)\/([0-9]+)/\2-\1-\3/'
```

The perl "s" command here uses a slash character to separate the parts of the command. That makes it a special character in this context, so for perl, the slash in the regular expression has to be escaped with a backslash. It also possible to use a different character than "/" in the "s" command. Using "=" instead, for example, the perl command would be:

```
cat data.txt | perl -pe 's=([0-9]+)/([0-9]+)/([0-9]+)=\2-\1-\3='
```

5. (*2 points*) The first item on each line of the file is a four-digit code for the academic term in which the course was taken. The first digit is always 1, the second and third digit give the last two digits of the year, and the fourth digit is 2 for Spring term, 4 for Summer term, or 6 for Fall term. All of the terms in the file are in the 21st century. You would like to have the terms listed in a more standard format such as Spring 2009, Summer 2018, and Fall 2015. What **three** regular expression search-and-replace operations could you use to make the change?

Answer:

We need to treat Spring, Summer, and Fall terms separately. They are distinguished by the last digit of the code. The changes can be made using the following search strings and replacement texts.

Search expression: `^1([0-9][0-9])2`
Replacement text: `Spring 20\1`

Search expression: `^1([0-9][0-9])4`
Replacement text: `Summer 20\1`

Search expression: `^1([0-9][0-9])6`
Replacement text: `Fall 20\1`

In the replacement text `^1([0-9][0-9])2`, the `^1` matches the 1 at the beginning of a line, and the group `([0-9][0-9])` captures the next two digits, which we need to use in the replacement text to represent the last two digits of the year. The final `2` in the search expression means that we are matching the code for a Spring term. In the replacement text `Spring 20\1`, the `\1` represents the captured group that holds the last two digits of the year.

6. (*2 points*) Each line of the file contains five fields, separated by commas. You would like to discard the last two fields, interchange the first two fields, and add spaces after the commas. So, for example, the line

1152,MATH 110,B-,23,5/29/2013

would be transformed to

MATH 110, 1152, B-

This change can be made using regular expression search-and-replace. You want to be careful to match the whole line and pull out the parts that you need. What search expression and replacement text could you use? Hint: .* can be useful here.

Answer:

There are five fields on each line, separated by commas. We can match each field with the regular expression `.*`, as long as we also include commas in the expression to match the commas that separate the fields. We need to match all five fields, but only the first three need to be in parenthesized groups, since they are the only ones that need to be copied into the output. In the replacement text, we just need to transpose the order of the first two groups, and use a comma and a space to separate the groups.

Search expression: `(.*),(.*),(.*),.*,.*`

Replacement text: `\2, \1, \3`

7. (*2 points*) The file includes lines with grades such as CR and W. You only want the data for letter grades. How could you use the *egrep* command to retain only the lines that contain letter grades? You need to find a regular expression that matches only those lines? A letter grade consists of one of the letters A, B, C, D, or F, optionally followed by a + or - sign. Some of the letters can occur in the file in other places besides letter grades, so be careful. Hint: Make use of certain commas in the file.

Answer:

Each grade is enclosed between two commas, and we can be sure to avoid unintended matches by including those commas in the search expression. Between the commas, we need to match one of the letter grades and an optional plus or minus sign. This can be done with

```
egrep ',[ABCDF](\+|-)?,' data.txt
```

8. (*2 points*) This problem does not use regular expressions. The fourth field of the file contains grades of various kinds such as A+, NC, and VW. You would like to know, without carefully reading the file, exactly what grades are possible. What command could be used on the command line to get a list of the grades that occur, without any repetition in the list? This can be done using a combination of the *cut* and *sort* commands. What would you add to the command to easily find out how many different possible grades there are?

Answer:

To get the third field from each line, where the fields are delimited by commas, we can use the command

```
cut -d "," -f 3
```

The output can be piped into `sort -u` to get a list of distinct entries from that field. If we want to know how many distinct entries there are, we can pipe that list into `wc -l`. So to see the list of distince grades, a command that will work is

```
cut -d "," -f 3 data.txt | sort -u
```

and to get the number of distinct grades,

```
cut -d "," -f 3 data.txt | sort -u | wc -l
```