Collaborating on homework is encouraged, but you must write your own solutions in your own words and list your collaborators. Copying someone else's solution will be considered plagiarism and may result in failing the whole course.

Please answer clearly and concisely. Explain your answers. Unexplained answers will get lower scores or even no credits.

(1) (18 points) Briefly justify (in about 3-6 sentences) the following statements.

(a) If L is regular, then

$$L' = \{w \in L \mid \text{the number of } \texttt{1}\text{'s in } w \text{ is divisible by } 3\}$$

is also regular.

(b) There are languages L_1 and L_2 such that L_2 is irregular and $L_1 L_2$ is regular.

(c) There is a regular language L such that

$$L^A = \{u \mid u \text{ is an anagram of } w \in L\}$$

is irregular.

Note: A string u is an anagram of a string w if u is obtained from w by rearranging the symbols. For example, `state` is an anagram of `taste`.

Formally, if $w = w_1 w_2 \cdots w_n$, then $u = w_{\sigma(1)} w_{\sigma(2)} \cdots w_{\sigma(n)}$ is an anagram of w, where σ is a permutation of $\{1, \cdots, n\}$.

(2) (32 points) Which of the following languages are regular, and which are not? To show a language is regular, give a DFA, NFA, or regular expression for it. To show a language is not regular, prove it using the pumping lemma or using pairwise distinguishable strings.

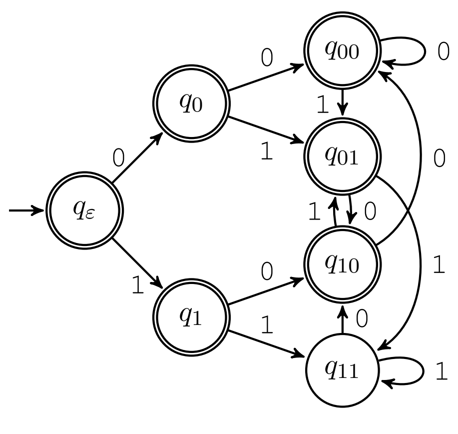
(a) $L_1 = \{w \in \{\texttt{0}, \texttt{1}\}^* \mid \text{The number of } \texttt{0}\text{'s and number of } \texttt{1}\text{'s in } w \text{ have the same parity}\}$
Two integers have the same parity if they are both odd or both even.

(b) $L_2 = \{ww^R \mid w \in \{\texttt{0}, \texttt{1}\}^*\}$
w^R denotes the string w written backwards, e.g. if $w = \texttt{war}$ then $w^R = \texttt{raw}$

(c) $L_3 = \{\texttt{0}^{n^4} \mid n \geqslant 0\}$

(d) $L_4 = \{u\#v \mid v \in \{\texttt{0}, \texttt{1}\}^* \text{ and } u \text{ is a prefix of } v\}$
A string u is a prefix of v if $v = ut$ for some string t, e.g. `eat` is a prefix of `eating`.

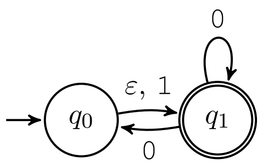
(3) (30 points) Consider the following DFA:



(a) Run the minimization algorithm on this DFA. Show the table of pairs of distinguishable states at the end of the algorithm (see page 22 of Lecture 7). Also draw the minimized DFA.

(b) Show that every pair of distinct states in the minimized DFA from part (a) is distinguishable (by giving a string to distinguish them, similar to page 10 of Lecture 7).

(c) Convert the following NFA into a regular expression using the conversion algorithm from class. Show the preprocessing step and how the NFA changes after each state is eliminated.



(4) (20 points) This is the only problem in this course where we deal with POSIX-style regular expressions, as opposed to regular expressions in formal language theory.

The file `propernames` contains a partial list of first names. Every line contains a name. Each name consists of one or more English letters in upper or lower case. To search for each of the following piece of information in the file, write a `grep` command of the form

`grep -iE '`*regex*`' propernames`

Also give a short explanation (1-3 sentences) how your regex works in each case.

Recall that the option `-i` ignores distinction between upper and lower case. Test your commands with the file `http://www.cse.cuhk.edu.hk/~siuon/csci3130/other/propernames`.

(a) Any name that ends with consecutive vowels or consecutive consonants, such as `Tan`**`ya`**.
`y` is regarded as a vowel.

(b) Any name with at most three consonants, so **`L`**`i`**`nd`**`e`**`r`** is excluded.

(c) Any name whose every even position is a consonant, such as `S`**`t`**`e`**`v`**`e`**`n`**.

(d) Any name containing some substring s of length two and the reversal of s, such that s and s^R do not overlap, such as **`Na`**`th`**`an`**

For part (d) you may want to use the backreference feature of `grep`.

Your solution should not use `-v` flag of `grep` which swaps non-matches with matches.