- Start as early as possible, and contact the instructor if you get stuck.
- See the course outline for details about the course's marking policy and rules on collaboration.
- Submit your completed solutions to **Crowdmark**.

1. Context-free and non-context-free languages

[4] (a) Prove that the language

$$L_a = \left\{w \in \{0,1\}^* \mid n_1(w) = n_0(w)^2\right\}$$

is **not** context-free. Recall that $n_1(w)$ denotes the number of occurrences of the symbol 1 in the string w, and $n_0(w)$ denotes the number of occurrences of the symbol 0 in the string w.

Proof. To prove the language L_a is **not** context-free we use the pumpimg lemma for context-free languages.

- Let $n > 0$ be arbitrary.
- Let $z = a^n b^{n^2}$, because $n_1(z) = n_0(z)^2$ we know $z \in L_a$.
- Consider the decompositions of z such that $z = uvwxy$ where,
  - $|vxw| \leq n$
  - $|vx| \geq 1$
  - $\forall i \in \mathbb{Z}^+ : uv^i wx^i y \in L_a$
- There are four cases to consider:
  1. When v and x are both equal to a^k for some $k \geq 0$
     - In this case, pumping i will increase the number of as in the string while the number of bs remain the same. This will lead to there being more a's than b's which contradicts the condition $n_1(z) = n_0(z)^2$.
  2. When v and x are both equal to b^k for some $k \geq 0$
     - In this case pumping i will increase the number of b's in the string without affecting the number of as. While the number of bs will remain higher than the number of as, each increase of i will cause a linear increment of b which is required to grow exponentially due to the condition $n_1(z) = n_0(z)^2$.
  3. When v is equal to a^k and x is equal to b^j for some $k, j \geq 0$
     - Let $v = a^k$ and $x = b^j$, where $k + j \geq 1$ and $k \leq n$, $j \leq n$.
     - The string $z = uvwxy$ will be $a^n b^(n^2)$, which has $n_a(z) = n$ and $n_b(z) = n^2$.
     - Now, if we pump down by selecting $i = 0$, we get $u(v^0)w(x^0)y = uwy$
     - Since v and x are non-empty, this means that u and y together will have some a's and b's, and w will have $n - k$ a's and $n^2 - j$ b's.
     - For this pumped string to be in L, we need $n_a(uwy) = n_b(uwy)^2$, but this won't be possible because $n_a(uwy)$ will not equal the square of $n_b(uwy)$ in general.
  4. When v is equal to b^k and x is equal to a^j for some $k, j \geq 0$
     - Let $v = b^k$ and $x = a^j$, where $k + j \geq 1$ and $k \leq n$, $j \leq n$.
     - The string $z = uvwxy$ will be $a^n b^(n^2)$, which has $n_a(z) = n$ and $n_b(z) = n^2$.

  - Now, if we pump down by selecting $i = 0$, we get $u(v^0)w(x^0)y = uwy$
  - Similar to the previous case, u and y together will have some a's and b's, and w will have $n^2 - k$ b's and $n - m$ a's
  - Again, for this pumped string to be in L, we need $n_a(uwy) = n_b(uwy)^2$, but this won't be possible because $n_a(uwy)$ will not equal the square of $n_b(uwy)$ in general.
- Since no decomposition of $z = a^n b^{n^2}$ can be pumped which contradict the pumping lemma.
- Therefore, by L_a is **not** context-free.

$\square$

[4] (b) Let $\Sigma = \{a, b, c\}$ be the alphabet for this part and for part 1c. Prove that the language

$$L_b = \left\{a^i b^j c^k \mid j \leq i \text{ or } k \leq i\right\}$$

is context-free.

Proof. To prove the language L_b is context-free we will define two context-free grammars G_1 and G_2 by listing their productions. We will show the union of these grammars to be equal to the language defined by L_b. As the class of context-free languages are closed under a finite union this will prove L_b to be context-free.

- Let $G_1 = (V_1, T_1, P_1, S_1)$ be the context free grammar with the start symbol S_1 defined by the following productions:

$$\begin{aligned} S_1 \to& S_1 c \mid T_1 \mid \varepsilon \\ T_1 \to& aT_1 b \mid aT_1 \mid \varepsilon \end{aligned}$$

  - We claim $L(G_1) = \{a^i b^j c^k | j \leq i\}$ by considering the productions in G_1.
    * We first consider the edge cases:
      · Starting at S_1, when $i = j = k = 0$ $S_1 \to \varepsilon$.
      · When $i = j = 0$ and k is arbitrary we have 0 a's and b's with an arbitrary number of cs. This can be obtained with the production $S_1 \to S_1 c$.
    * Next we consider the general case where i, j, k are arbitrary.
      · By taking the production $S_1 \to S_1 c$ k times we can always include k c's in our string
      · To add a's and b's to the string we must take the production $S_1 \to T_1$.
      · For T_1 we can add a balanced number of a's and b's with the production $T_1 \to aT_1 b$
      · At any time we can take production $T_1 \to aT_1$ to add additional a's to the string
    * We note that any number of productions of T_1 will lead to either the case where there are a balanced number of a's and b's or a case where there are more a's than b's.
    * Furthermore, at any time we have shown the production path to add an arbitrary number of c's
    * As there are no more productions to consider in G_1, the condition $j \leq i$ for $a^i b^j c^k$ will always hold.
    * Thus $L(G_1) = \{a^i b^j c^k | j \leq i\}$
- Let $G_2 = (V_2, T_2, P_2, S_2)$ be the context free grammar with the start symbol S_2 defined by the following productions:

$$S_2 \to aS_2 \mid aS_2 c \mid S_2 c \mid \varepsilon$$

  - We claim $L(G_2) = \{a^i b^j c^k | k \leq i\}$ by considering the productions in G_2.
    * We first consider the edge cases:
      · Starting at S_2, when $i = j = k = 0$ $S_2 \to \varepsilon$.

- When $i = k = 0$ and j is arbitrary we have 0 a's and c's with an arbitrary number of bs. This can be obtained with the production $S_2 \to S_2 b$.

* Next we consider the general case where i, j, k are arbitrary.
  - By taking the production $S_2 \to S_2 b$ j times we can always include j b's in our string
  - To add a's and c's to the string we must take the production $S_2 \to aS_2c$ or $S_2 \to aS_2$.
  - We add a balanced number of a's and c's at any time with the production $S_2 \to aS_2c$
  - At any time we can take production $S_2 \to aS_2$ to add additional a's to the string
* We note that any number of productions of S_2 will lead to either the case where there are a balanced number of a's and c's with an arbitary number of b's or a case where there are more a's than c's.
* As there are no more productions to consider in G_2, the condition $k \leq i$ for $a^ib^jc^k$ will always hold.
* Thus $L(G_2) = \{a^ib^jc^k | k \leq i\}$

- We have shown that $L(G_1) = \{a^ib^jc^k | j \leq i\}$ and $L(G_2) = \{a^ib^jc^k | k \leq i\}$.
- It is clear that $L(G_1) \cup L(G_2) = \{a^ib^jc^k | j \leq i \text{ and } k \leq i\}$
- As G_1 and G_2 are defined by context-free grammars they must describe context-free languages.
- Since the class of context-free languages are closed under union and $L(G_1) \cup L(G_2) = L_b$
- Therefore, L_b is a context-free language.

$\square$

[4] (c) Prove that the complement, L_b', is **not** context-free. (**Remark:** This proves that L_b is **not** a DCFL.)

2. Closure rules for CFLs

[2] (a) Let L be a CFL and let F be a finite language. Prove that $L \backslash F = \{w \in L \mid w \notin F\}$ is a CFL.

Proof.

- $L \setminus F \underset{DeMorgan}{=} L \cap F^c$
- F is finite and thus a regular language.
- As regular languages are closed under complement F^c is regular.
- The class of context-free languages are closed under a finite interection with a regular language.
- As L is a context-free language by definition $L \cap F^c$ is a context-free language.
- Since $L \cap F^c$ is context free, therefore $L \setminus F$ is a context-free language.

$\square$

[2] (b) Let L be a **non**-context-free language and let F be a finite language. Prove that $L \setminus F = \{w \in L \mid w \notin F\}$ is a **non**-context-free language.

Proof.

- We know $L \setminus F \underset{DeMorgan}{=} L \cap F^c$
- F is finite and thus a regular language.
- Assume $L \setminus F$ is a context-free language.
- We know context-free languages are closed under intersection with a regular language.
- Since $L \setminus F$ is a context-free language then L must be a context-free language but this is a contradiction to the definition of L
- Therefore $L \setminus F$ is **not** context-free.

$\square$

[2] (c) Let L be a **non**-context-free language and let F be a finite language. Prove that $L \cup F$ is a **non**-context-free language.

Proof.
- Let $L = \{a^i b^i c^i | i \geq 0\}$ which we know is not context-free.
- Define $M = L(a*)$ which is regular since it is defined by a regular expression.
- Note, $L \cup M = \{a^i b^i c^i | i \geq 0\}$ which is equivallent to the language L
- Since L is not context-free and $L \cup M$ is equivallent to L
- Therefore, $L \cup F$ is not context free.

$\square$

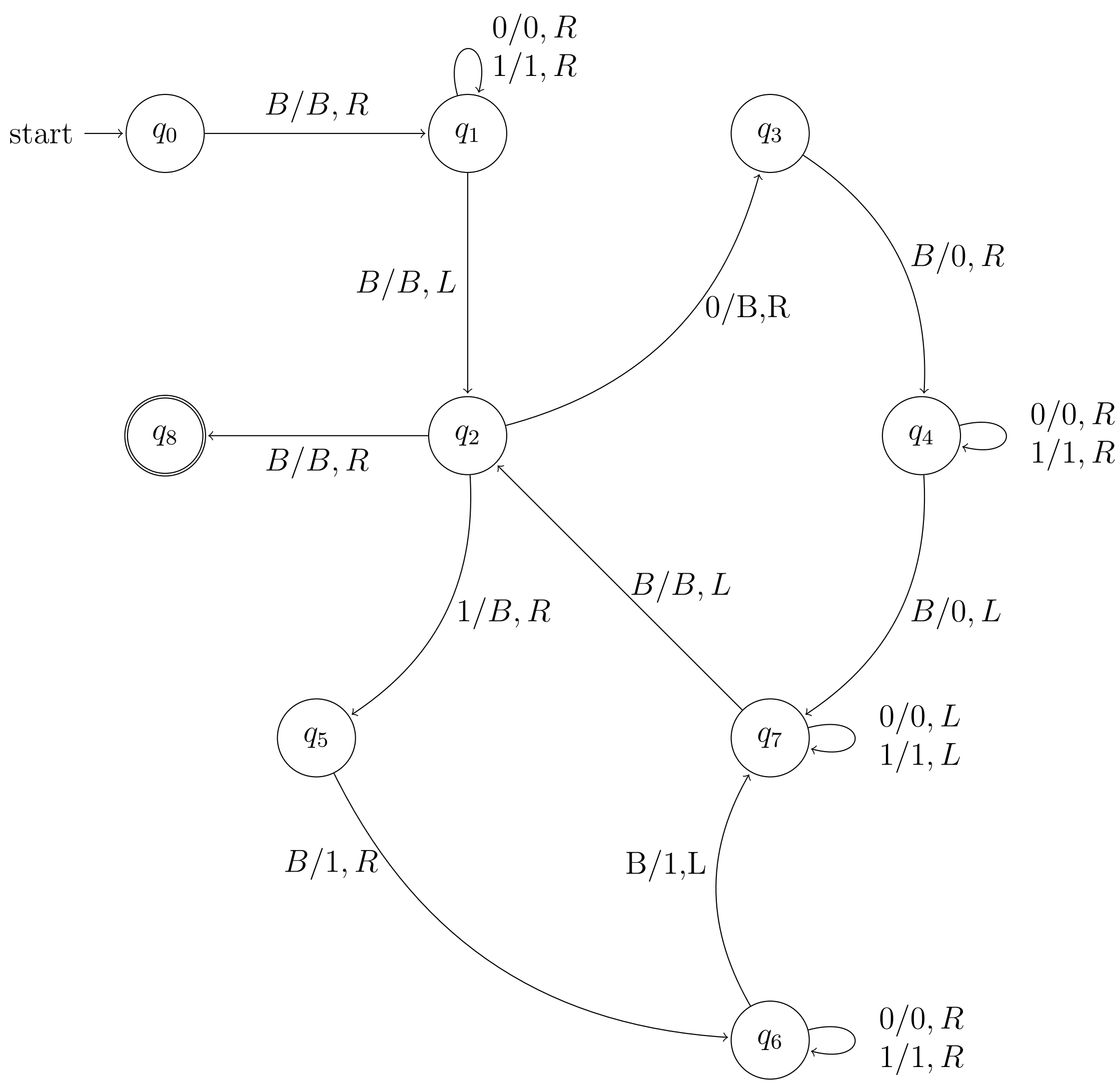
3. Computations in a Turing machine

Let M be a Turing Machine over the alphabet $\Sigma = \{0, 1\}$. Let M's tape alphabet be $\{0, 1, B\}$. Let M's states be $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$, with q_8 being the sole final state. Let the transition function, δ, for M be defined by the following table.

q	x	$\delta(q,x)$	q	x	$\delta(q,x)$	q	x	$\delta(q,x)$
q_0	B	(q_1, B, R)	q_2	B	(q_8, B, R)	q_6	0	$(q_6, 0, R)$
q_1	0	$(q_1, 0, R)$	q_3	B	$(q_4, 0, R)$	q_6	1	$(q_6, 1, R)$
q_1	1	$(q_1, 1, R)$	q_4	0	$(q_4, 0, R)$	q_6	B	$(q_7, 1, L)$
q_1	B	(q_2, B, L)	q_4	1	$(q_4, 1, R)$	q_7	0	$(q_7, 0, L)$
q_2	0	(q_3, B, R)	q_4	B	$(q_7, 0, L)$	q_7	1	$(q_7, 1, L)$
q_2	1	(q_5, B, R)	q_5	B	$(q_6, 1, R)$	q_7	B	(q_2, B, L)

Let M begin processing in the configuration $(q_0, \underline{B}w)$, where $w \in \Sigma^*$ is the input word.

[4] (a) Draw a diagram for M.



[2] (b) Give the sequence of instantaneous descriptions of M as it processes the input word $w = 01$.

$$
\begin{aligned}
(q_0, \underline{B}01) &\vdash (q_1, \underline{0}1) \\
&\vdash (q_1, 0\underline{1}) \\
&\vdash (q_1, 01\underline{B}) \\
&\vdash (q_2, 0\underline{1}B) \\
&\vdash (q_5, 0B\underline{B}) \\
&\vdash (q_6, 0B1\underline{B}) \\
&\vdash (q_7, 0B\underline{1}1) \\
&\vdash (q_7, 0\underline{B}11) \\
&\vdash (q_2, \underline{0}B11) \\
&\vdash (q_3, B\underline{B}11) \\
&\vdash (q_4, B0\underline{1}1) \\
&\vdash (q_4, B01\underline{1}) \\
&\vdash (q_4, B011\underline{B}) \\
&\vdash (q_7, B01\underline{1}0) \\
&\vdash (q_7, B0\underline{1}10) \\
&\vdash (q_7, B\underline{0}110) \\
&\vdash (q_7, \underline{B}0110) \\
&\vdash (q_2, \underline{B}B0110) \\
&\vdash (q_8, \underline{B}0110)
\end{aligned}
$$

[4] (c) Give the sequence of instantaneous descriptions of M as it processes the input word $w = 100$.

$$
\begin{aligned}
(q_0, \underline{B}100) &\vdash (q_1, \underline{1}00) \\
&\vdash (q_1, 1\underline{0}0) \\
&\vdash (q_1, 10\underline{0}) \\
&\vdash (q_1, 100\underline{B}) \\
&\vdash (q_2, 10\underline{0}) \\
&\vdash (q_3, 10B\underline{B}) \\
&\vdash (q_4, 10B0\underline{B}) \\
&\vdash (q_7, 10B\underline{0}0) \\
&\vdash (q_7, 10\underline{B}00) \\
&\vdash (q_2, 1\underline{0}B00) \\
&\vdash (q_3, 1B\underline{B}00) \\
&\vdash (q_4, 1B0\underline{0}0) \\
&\vdash (q_4, 1B00\underline{0}) \\
&\vdash (q_4, 1B000\underline{B}) \\
&\vdash (q_7, 1B00\underline{0}0) \\
&\vdash (q_7, 1B0\underline{0}00) \\
&\vdash (q_7, 1B\underline{0}000) \\
&\vdash (q_7, 1\underline{B}0000) \\
&\vdash (q_2, \underline{1}B0000) \\
&\vdash (q_5, B\underline{B}0000) \\
&\vdash (q_6, B1\underline{0}000) \\
&\vdash (q_6, B10\underline{0}00) \\
&\vdash (q_6, B100\underline{0}0) \\
&\vdash (q_6, B1000\underline{0}) \\
&\vdash (q_6, B10000\underline{B}) \\
&\vdash (q_7, B1000\underline{0}1) \\
&\vdash (q_7, B100\underline{0}01) \\
&\vdash (q_7, B10\underline{0}001) \\
&\vdash (q_7, B1\underline{0}0001) \\
&\vdash (q_7, B\underline{1}00001) \\
&\vdash (q_7, \underline{B}100001) \\
&\vdash (q_2, \underline{B}B100001) \\
&\vdash (q_8, \underline{B}100001)
\end{aligned}
$$

[2] (d) Briefly describe the algorithm which M performs, given any input word $w \in \Sigma^*$.

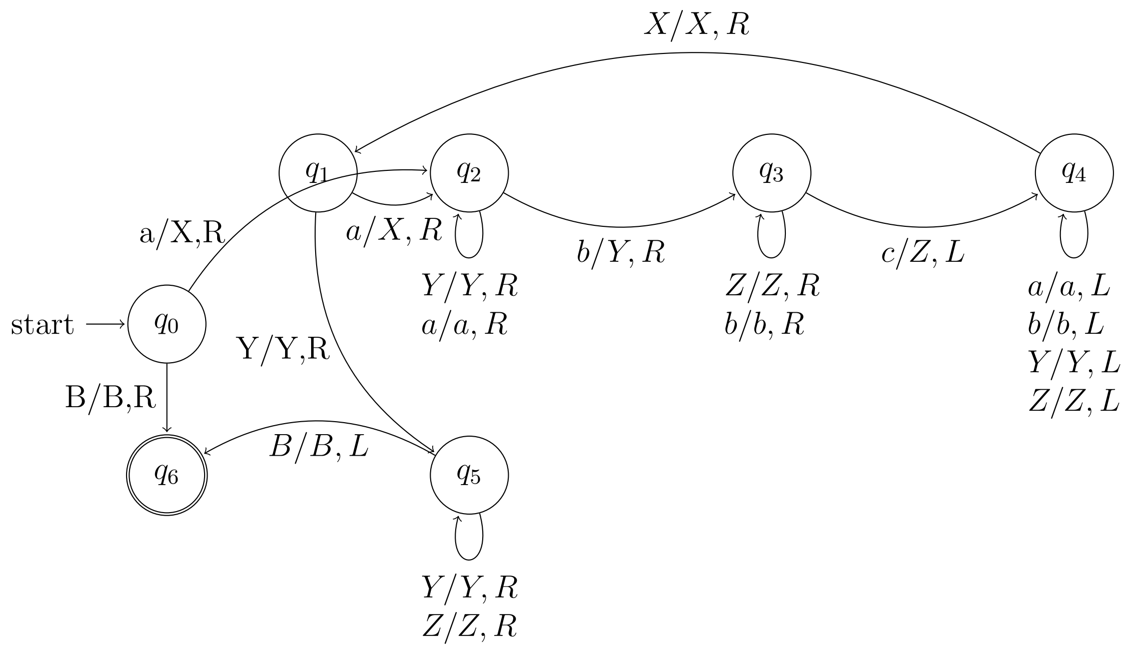
The Turing Machine M takes an input word $w \in \Sigma^*$ and returns $x \in \Sigma^*$ such that $x = ww^R$. In plain english the Turing Machine takes an input word and returns the concatenation of the input word followed by it's reversal.

4. A language which is recursive but not context-free

[8] Let $\Sigma = \{a, b, c\}$. Recall from the lectures that this language is **not** context-free:

$$L = \left\{ a^i b^i c^i \mid i \geq 0 \right\}.$$

Give an **algorithm** for a Turing machine that **decides** membership in the language L. **You do not need to give a detailed diagram for the Turing machine**, provided you describe your algorithm clearly enough. Argue informally why your algorithm is correct.



- Add the symbols X, Y, Z, a, b, B to the stack alphabet
- The input word is loaded on the tape and the head is positioned at the first symbol.
- If the tape is blank, accept
- If the tape is not blank the TM will loop trough the following steps:
  - Read an a replacing it with an X and move the tape head to the right.
  - Keep moving right skipping any a or Y symbols until a b is found.
  - Read the b and replace it with a Y.
  - Move right for any b or Z symbols until a c is read.
  - When a c is read replace it with a Z and move the tape head left.
  - Keep moving the tape head left for any symbol (except blank) until an X is found. This X is in the position of the last a that was replaced in this loop.
  - While the character after this X is a continue looping
- The above loop will replace an a with an X, a b with a Y and a c with a Z at each iteration.
- If there are an unbalanced number of a's b's and c's the machine will crash as there will be no path to take.
- Only when the machine sees a Y following an X does it move right through any Y's and Z's on the tape until a blank is reached.

- If any character is seen before the blank the machine will crash as the number of a's b's and c's are unbalanced.
- When the expected blank is read the machine immediatly accepts.

5. Every context-free language is recursive

[6] Let Σ be a non-empty finite alphabet. Let G be an arbitrary context-free grammar over Σ, and let $L = L(G)$. Give an algorithm for a Turing machine which decides membership in the language L. Argue informally why your algorithm is correct.