For the midterm (on Thursday), you are allowed to bring in a handwritten cheat sheet.

$L = \{a^n b^n c^n \mid n \geq 0\}$ is a non-regular, non-CFL language, however, it can be decided by a Turing Machine.

Let's give a high-level description of a TM for $L$:

"on input $w$:

1. If $w$ is $\epsilon$, accept
   
2. If $w$ starts with "b" or "c", reject

3. Cross out an "a", and skip over "a"'s and "b"'s to find a "b"
   
4. Cross out a "b", skip over b's and "c"'s to find a "c"
   
5. Cross out "c" and go back to the beginning of the tape (looking for "a" or "b"
   
6. Go forward, make sure everything is crossed out
   
If find "b" or "c", reject

else accept"
92: just crossed out an "a", skip over remaining a's and y's to get to a "b" to cross out.
93: just crossed out a "b".
94: just crossed out a "c", go backwards on the tape until we find either an "a" or an "x".
95: have crossed out all "a"'s, check if there are remaining "b"'s or "c"'s. etc.

Note II: The original low-level implementation of this step (3) was incorrect.

Main takeaways:
High-level descriptions of Turing Machines are very useful and usually nice and clean. Low-level (i.e., transition function or diagram) are complicated and hard to make. In general we want you to give us high-level descriptions.
Definition: A Turing Machine (TM) is a 7-tuple \((Q, \Sigma, \Gamma, S, q_0, q_{accept}, q_{reject})\)

- \(Q\) - set of states
- \(\Sigma\) - the input alphabet \(\sigma \not\in \Sigma\)
- \(\Gamma\) - the tape alphabet \(\text{true} \in \Gamma\), \(\Sigma \subset \Gamma\)
- \(S\) - transition function \(S: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\)
- \(q_0\) - start state
- \(q_{accept}\) - accept state. TM accepts once it hits \(q_{accept}\)
- \(q_{reject}\) - reject state. TM rejects once it hits \(q_{reject}\)

A configuration of a TM:

```
| a | b | b | a | b |
```

\(q_f\) current state

- The start configuration of a TM \(M\) on input \(w\) is \((q_0, w)\)
- The configuration \(C = (q_0, w)\) yields \(C' = (q_1, a, c, v)\)
- \(C'\) yields \((q_2, v)\) if \(S(q_1, a) = (q_2, c, L)\)
- \(C'\) yields \((q_3, v)\) if \(S(q_1, b) = (q_3, c, R)\) (rejecting)

M accepts \(w\) if \(\exists\) a sequence \(C_1, \ldots, C_k\) such that \(C_1\) is the start configuration of \(M\) on \(w\), \(C_k\) is an accepting configuration.

C is accepting if \(C = (q_{accept})\) for any strings \(w, v\).

Note: the TM is deterministic. It is like a more complex DFA, as it always makes one decision for each input, state.

Definition: \(L(M) = \{w \mid M\text{ accepts } w\}\)
Church-Turing Thesis:

A Turing Machine is a universal model of computation. 

ie. for any L recognized by your favourite programming language, 

\exists a TM M that recognizes L.

An important result of the CT thesis is that variations of the TM are not more powerful than the classic TM.

For example,

- \( \Gamma = \Sigma \cup \{ \text{L}\} \) because we can encode the original \( \Gamma \) with multiple characters from \( \Sigma \) boxes for each character in \( \Gamma \).

- "stay-put" TM: if we allow the TM to go Left, Right, or Stay, i.e., on any input. This is equivalent to a classical TM as we can simulate "stay" by LR.

- "doubly-infinite" TM: if we allow the TM to have a tape that extends infinitely in both directions:

  ![Doubly Infinite Tape](image)

which a TM with a singly infinite tape can be simulated by a regular TM.

Let M be a TM in a doubly infinite tape. To simulate M using a regular TM:

- On input w:
  - move the start position of the tape
  - run M on input w over the singly infinite tape.
  - If M ever attempts to go left from the marked position:
    - move content of tape one position to the right, overwrite position with a 0.
    - go back to the marked blank position. Continue to run M on the tape.