1. For each of the following languages, determine whether it is context-free or not, if it is, give a context-free grammar for it; if not, prove it.

1.1) $L = \{\omega \in \{0, \#\}^\star \mid \omega\ has\ 0\verb|^|n \# 0\verb|^|2n \# 0\verb|^|3n\ pattern,\ where\ n \geq 0\}$

**The above language is nonregular because a DFA/NFA/regular expression cannot be created. In order to determine if the language is context-free or not, use pumping lemma. We first assume that L is context-free, then according to the pumping lemma, there must exist a pumping length p, such that for any string w, w is in the set of L. w >=p, w can be written as w = u*v*x*y*z. It also satisfies the 3 conditions:**

- **|vxy| <= p**
- **|vy| > 0 so v and y cannot both be empty**
- **A new string u*(v^i)*x*(y^i)*z is also in the language**

**We can now pick a special string to perform our test cases. S =0^p # 0^2p # 0^3p. S will be in the language as long as the number of 0's between the #'s stays in the ratio (1:2:3). If s no longer follows this ratio then it is no longer in the language. S should still be in the string if a new string = u*(v^i)*x*(y^i)*z where i can be pumped up and down. In order to deal with the cases I split the string into 3 segments and refer to each segment separated by #'s by a, b, and c.**

**Case 1 & 2 & 3:**

- **Case 1 is if v and y are only in the A segment of the string. If this is the case pumping up would lead to there being a couple more zeros in the a segment of the string. This will cause the (1:2:3) ratio to no longer hold. Case 2 is if v and y are only in the B segment of the string. In this case you could be pumping the middle segment of zero which would no longer hold the 1:2:3 ratio. Case 3 is if v and y are only in the C segment of the string. In this case, pumping up or down would also lead to the 1:2:3 ratio of zeros to no longer hold. This means that all 3 cases fail and are not proven**

**Case 4 & 5:**

- **Case 4 is if v and y are partly in the A segment of the string and the B segment of the string. Pumping up, in this case, would lead to both the A zeros to increase as well as the B zeros to increase. This would lead to a breaking of the 1:2:3 ratio. Case 5 is if v and y are partly in the B segment and the C segment of the string. Pumping up in this case would lead to both the B zeros to increase as well as the C zeros to increase. This would also lead to a breaking of the 1:2:3 ratio.**

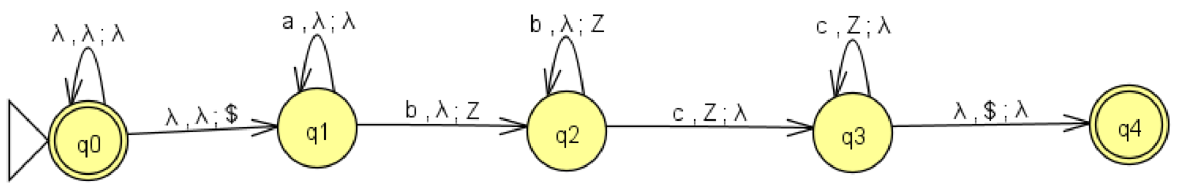
1.2) $L = \{x \cdot y \mid x, y \in \{0, 1\}^\star\ and\ |x| = |y|,\ but\ x \neq y\}$

**The above language is Context free because Context Free Grammar can be made for the language. The Context-Free Grammar would be the following:**

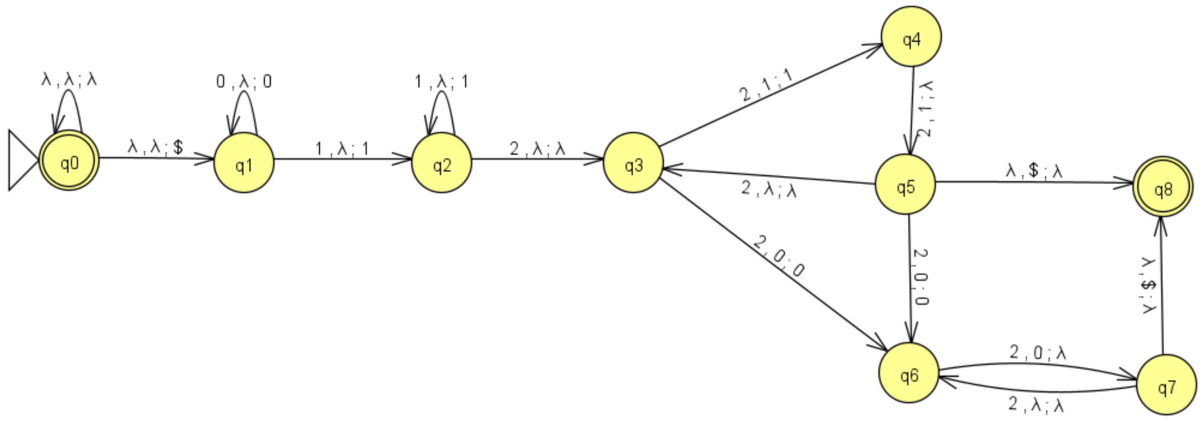
- **L -> MN | NM**
- **M -> 0 | PMP**
- **N -> 1 | PNP**
- **P -> 1 | 0**

2. Draw the state machine of a deterministic PDA for the following languages (assume all missing transitions go to a dead state; you do not have to draw those transitions). 10 pts each.

2.1) $L = \{a^\wedge i\ b^\wedge j\ c^\wedge k \mid j = k$ and $i, j, k \geq 0\}$



2.2) $L = \{0^\wedge(2x)\ 1^\wedge(3y)\ 2^\wedge(x+y) \mid x, y \geq 0\}$



3. Give the sequence of configurations that the specified machine below enters when started on the indicated input string.

Q IS BEHIND THE CHARACTER ITS ON

3.1) # on M1 (see Turing machine M1 from our textbook pp. 166 Figure 3.10).

- **(q1)#**
- **#(q8)**
- **#□(qAccept)**
- **ACCEPTED**

3.2) 101#010 on M1 (see Turing machine M1 from our textbook pp. 166 Figure 3.10).

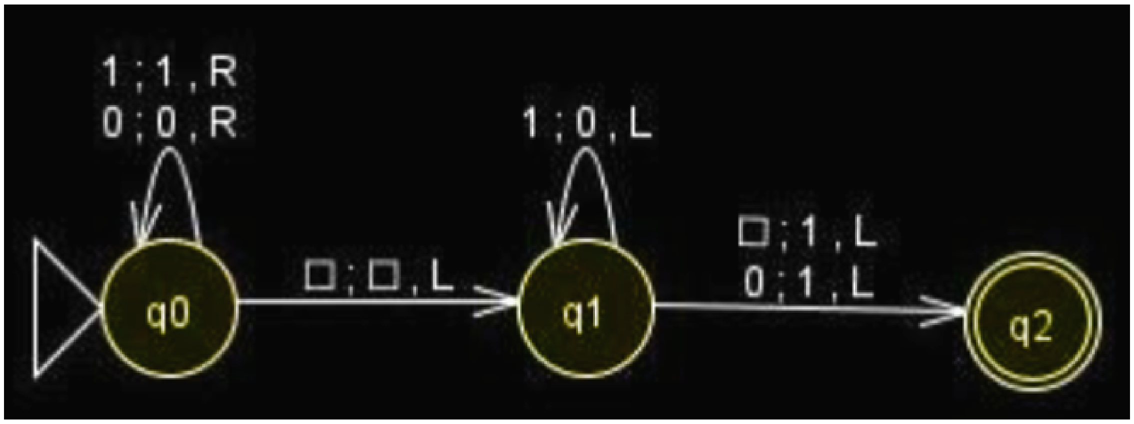
- **(q1)101#010**
- **x(q3)01#010**
- **x0(q3)1#010**
- **x01(q3)#010**
- **x01#(q5)010**  **NO Q5 w/ input 0 transition**
- **REJECTED**

3.3) 00 on M2 (see Turing machine M2 from our textbook pp. 165 Figure 3.8).

- **(q1)00**
- **□(q2)0**
- **□x(q3)□**
- **□(q5)x□**
- **(q5)□x□**
- **□(q2)x□**
- **□x(q2)□**
- **□x□(qAccept)**
- **ACCEPTED**

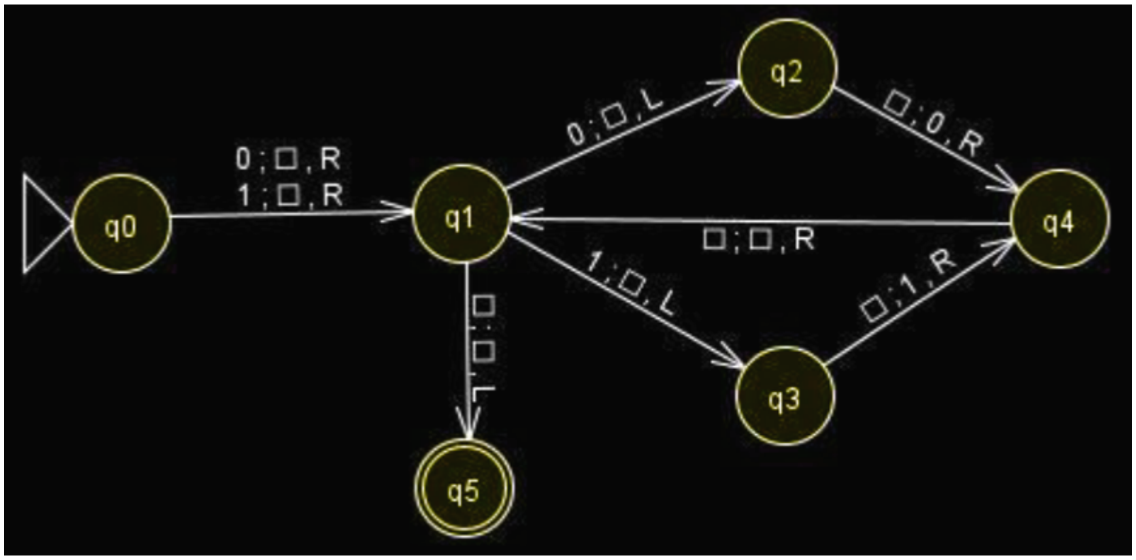
4. Given the following Turing machine's state diagram, assume Σ = {0, 1}, and describe in English what the TM computes.

4.1)



- **Reads the input as a binary number and adds one to it**

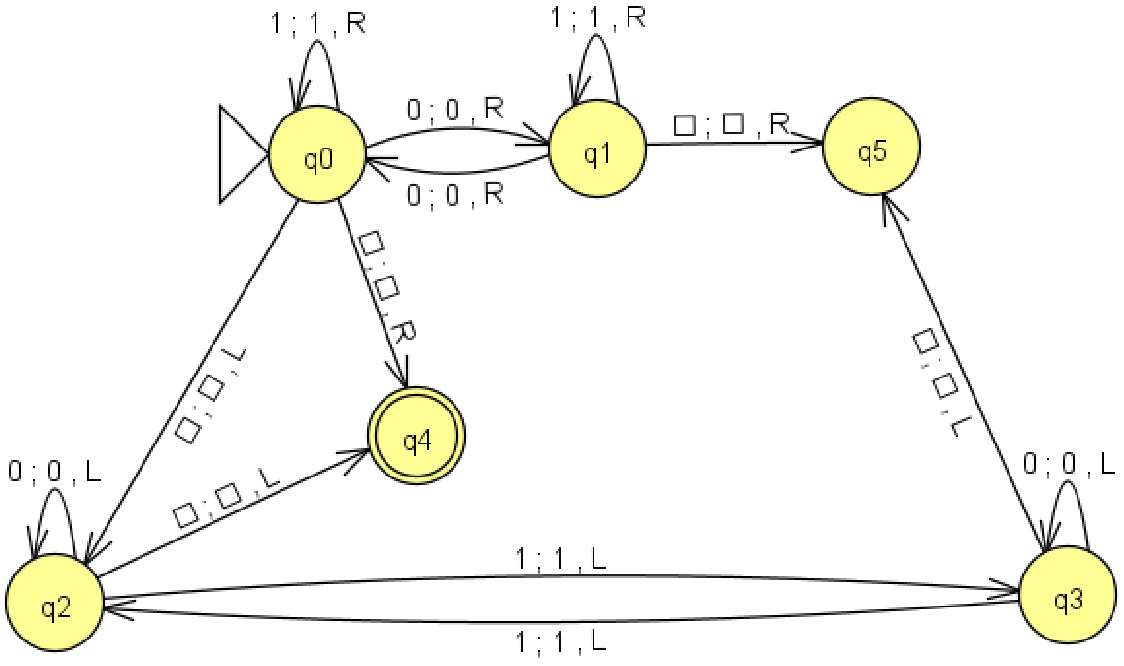
4.2)



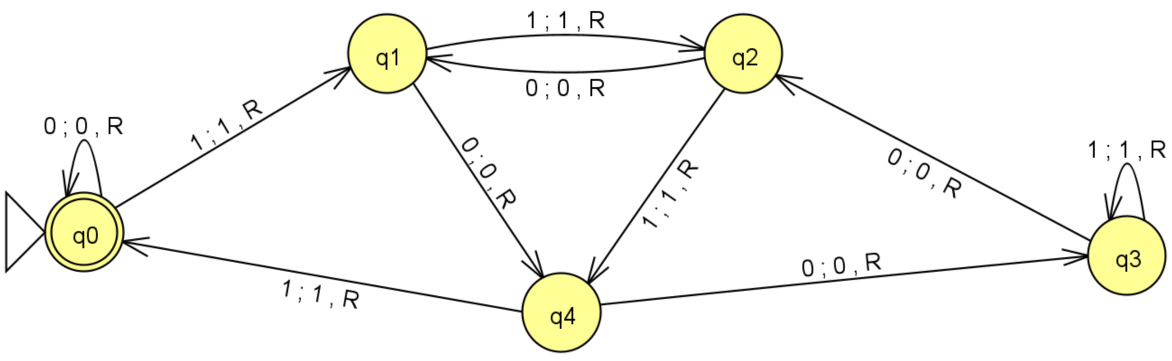
- **Shifts the binary input once over to the left by one**

5. Give the formal description of a Turing machine, (i.e. list the 7 tuples, specifically draw its state diagram) that decides each of the following languages.

5.1) $L = \{\omega \in \{0, 1\}^\star \mid \omega \textit{ has even number of both } 0 \textit{ and } 1\}$



5.2) $L = \{\omega \in \{0, 1\}^\star \mid \textit{if intepret } \omega \textit{ as decimal number, it's divisable by } 5\}$. i.e. L = {0, 101, 1010, 1111, ...}



6. Give the implementation-level description (English prose description of how the tape head moves and what is written to the tape) of the Turing machine that decides the following language.

Σ = {$a$, $b$, $c$, $d$}
$L$ = {$a$^$m$ $b$^$n$ $c$^$m$ $d$^$n$ | $m$, $n$ > 0}

**<u>Implementation level design description:</u>**

1. **Start at q0 andgo to read all the a's and go R**
2. **When you get to the b's you skip R on all of them till you hit the c's**
3. **Once you hit the c's transition to another state and replace the first c with some character "y"**
4. **Go L and skip all the b's until you get to the first a on the string and replace the character with some character "x"**
5. **Now go all the way to the R and skip all the b's till you get to the first c character and replace it with character "y"**
6. **Repeat this bouncing between a and c until there are no more of either both of the characters or one of the characters**
7. **Once you reach the last c character and replace it with a y, go all the way L and check if there is an a to replace to complement the c that has been replaced, if there isn't, then there are more c's than a's.**
8. **After you replace the last c or a, you go and check the other letter and see if there are any more left and if there are then that means the a's are not equal to the c's and go directly to a rejecting state**
9. **If the a's and c's are equal then you move to a new state and do steps 2-8 with the b's and d's.**
10. **Instead of skipping the b's in this case, you must skip the c's, bounce R from the b's, and bounce L from the d's.**
11. **You can replace b and d with some variable for each bounce but they must be different from the characters used for a and b and they can not be the same.**
12. **If the same conditions are met for b and d then you go to the accepting state. Otherwise, you go to the rejecting state.**