1. Define NFA? 2
2. For the grammar E → E+E, E→E\*E, E→id, construct a parse tree for the string id\*id\*id+id.
3. State Pumping lemma for regular languages
4. Write about Leftmost derivation and rightmost derivation with example?
5. Define context free language?
6. Mention ID format of Turing Machine
7. Give an example for undecidable problem.
8. List and explain the components of finite state automata and explain the advantages of Finite State Machine?
9. Design DFA for the following over {0, 1}.

(i)All string containing not more than three 0’s.

(ii)All strings that has at least two occurrences of 1 between any two occurrences of 0

1. Show that the language L={ww R | w € {a,b}\* } is generated with context free grammar?
2. Construct a CFG for palindrome. Generate (i) 0110110 (ii) 010010
3. Construct a PDA for L={a n b n c n /n &gt;0}
4. Construct a PDA for L={wcw R /w €(0+1)\*}
5. Explain the programming techniques for construction of Turing Machine and Explain the TM as Enumerator
6. Design a Turing machine for L={ww R | w€(0+1)\*,R stands for reverse
7. Explain the process of converting CNF to GNF with an example.
8. Explain about Universal Turing Machine



1. **Differentiate between NFA and DFA**
2. **Define Push Down Automata.**
3. **Bring out differences between Moore and Mealy machines?**
4. **What is ambiguous grammars? Give an example.**
5. **What are the components of a Turing Machine?**
6. **What is Universal Turing Machine**
7. **Construct a DFA for the Regular Language consisting of any number of a’s and b’s**
8. **Explain about the Closure Properties of Regular sets?**
9. **Define Ambiguous Grammar? Check whether the grammar S→aAB, A→bC/cd,,C → cd, B →c/d Is Ambiguous or not?**
10. **Explain in detail about Melay and Moore Machines with an example.**
11. **Show that L={a P /p is prime} is Context free?**
12. **Construct a DFA accepting the set of all strings ending with 00?**
13. **explain the decision properties of CFL**
14. **Construct Turing machine for the languages containing the set of all strings of balanced paranthesis?**
15. **Define Turing Machine? Explain about the Model of Turing Machine?**
16. **Explain in detail about Chomsky’s hierarchy**
17. **Explain in detail about PCP?**

Answers:

In place of a and b take 0 and 1

**DPDA for wcwR w ε (a,b)\***

Some string will come followed by one 'c', followed by reverse of the string before 'c'.  
So we get to know that 'c' will work as an alarm to starting poping STACK.  
So we will pop every 'a' with 'a' and every 'b' with 'b'.  
For every two a's and b's push them into STACK  
When 'c' comes do nothing.  
Starting poping STACK: 'a' for 'a' and 'b' for 'b'.  
We have designed the PDA for the problem:

STACK Transiton Function

δ(q0, a, Z) = (q0, aZ)

δ(q0, a, a) = (q0, aa)

δ(q0, b, Z) = (q0, bZ)

δ(q0, b, b) = (q0, bb)

δ(q0, a, b) = (q0, ab)

δ(q0, b, a) = (q0, ba)

// this is decision step

δ(q0, c, a) = (q1, a)

δ(q0, c, b) = (q1, b)

δ(q1, b, b) = (q1, ε)

δ(q1, a, a) = (q1, ε)

δ(q1, ε, Z) = (qf, Z)

**Note: *qf is Final State***

Explanation

Lets see, how this DPDA is working:  
We will take one input string: **"abbcbba"**

 Scan string from left to right

 First input is 'a' and follow the rule:

 on input 'a' and STACK alphabet Z, push the two 'a's into STACK as : (a,Z/aZ) and state will be q0

 Second input is 'b' and so follow the rule:

 on input 'b' and STACK alphabet 'a', push the 'b' into STACK as : (b,a/ba) and state will be q0

 Third input is 'b' and so follow the rule:

 on input 'b' and STACK alphabet 'b', push the 'b' into STACK as : (b,b/bb) and state will be q0

 Fourth input is 'c' and so follow the rule:

 on input 'c' and STACK alphabet 'a' or 'b' and state q0, do nothing as : (c,b/b) and state will be q1

 Fifth input is 'b' and so follow the rule:

 on input 'b' and STACK alphabet 'b' (state is q1), pop one 'b' from STACK as : (b,b/ε) and state will be q1

 Sixth input is 'b' and so follow the rule:

 on input 'b' and STACK alphabet 'b' (state is q1), pop one 'b' from STACK as : (b,b/ε) and state will be q1

 Seventh input is 'a' and so follow the rule:

 on input 'a' and STACK alphabet 'a' and state q1, pop one 'a' from STACK as : (a,a/ε) and state will remain q1

 We reached end of the string, so follow the rule:

 on input ε and STACK alphabet Z, go to final state(qf) as : (ε, Z/Z)

10.

Construct a context-free grammar G such that L(G) = {wcwR : w ∈ {a, b} ∗ }. Justify your answer.]

Solution Σ = {a, b, c}, V = Σ ∪ {S}, R = {S → aSa|bSb|epsilon

take one example of string and derive by using above productions

11.Construct a CFG for palindrome. Generate (i) 0110110 (ii) 010010

**S → 0S0 | 1S1 | 0 | 1 | ϵ**.

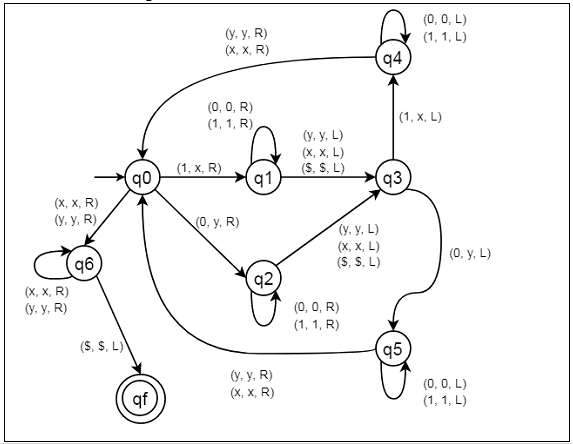
Above stings generate it by using above production

# Construct a Turing Machine for language L = {wwr | w ∈ {0, 1}}

Here we will see how to make a Turing machine for language L = {WWr |W belongs to {0, 1}}. So this represents a kind of language where we will use only two characters 0s and 1s. The w is a string and wr is reverse of it. So if w = 10110, then wr will be 01101. So the Turing machine will accept the string z = 1011001101.

To solve this, we will use this approach. First check the first symbol, if it’s 0 then replace it using y and if that is 1, then replace using x. Then go to the end of string. So last symbol is same as the first one. We replace it also by x or y depending on it. After that again come back to the position next to the symbol replace from the starting and repeat the same process that is mentioned above. We have to keep in mind that since wr is reverse of w of both of them will have equal number of symbols. Every time replace a nth symbol from beginning of string, replace a corresponding nth symbol from the end.

## State Transition Diagram



12.

## Problem

Construct deterministic push down automata (DPDA) for anbn where n>=1.

## Solution

So, the strings which are generated by the given language are as follows −

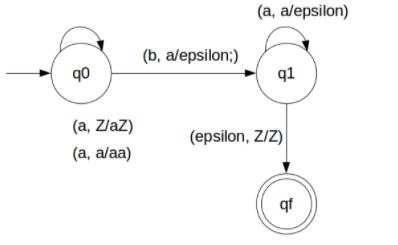
L={ab,aabb,aaabbb,….}

That is we have to count equal number of a’s and b’s

This can be achieved by pushing a's in STACK and then we will pop a's whenever "b" comes.

Finally at the end of the strings if nothing is left in the STACK then we can declare that language is accepted in the PDA.

The transition diagram is as follows −



## Transition Functions

The transition functions are as follows −

δ(q0, a, Z) = (q0, aZ)

δ(q0, a, a) = (q0, aa)

δ(q0, b, a) = (q1, ε)

δ(q1, b, a) = (q1, ε)

δ(q1, ε, Z) = (qf, Z)

## Explanation

**Step 1** − Let’s take an input string: "aabb".

**Step 2** − Scan string from left to right.

**Step 3** − First input is 'a' and the rule.

     input 'a' and STACK alphabet Z, then

     push the input 'a' into STACK : (a,Z/aZ) and state will be q0.

**Step 4** − Second input is 'a' and the rule.

     input input 'a' and STACK alphabet 'a', then

     input push the input 'a' into STACK : (a,a/aa) and state will be q0.

**Step 5** − Third input is 'b' and the rule.

     input 'b' and STACK alphabet 'a', then

     pop STACK with one 'a': (b,a/&epsiloon;) and state will be now q1.

**Step 6** − Fourth input is 'b' and the rule.

     input 'b' and STACK alphabet 'a' and state is q1, then

     pop STACK with one 'a' : (b,a/&epsiloon;) and state will be q1.

**Step 7** − We reached the end of the string, the rule.

     input ε and STACK alphabet Z, go to final state(qf) : (ε, Z/Z)