**Q1.what is finite automata?applications of finite automata? Minimal dfa that constructs a’s and b’s string where each string stars and ends with same symbol?**

* Ans. Finite automata are used to recognize patterns.
* It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.
* At the time of transition, the automata can either move to the next state or stay in the same state.
* Finite automata have two states, **Accept state** or **Reject state**. When the input string is processed successfully, and the automata reached its final state, then it will accept.

The **Applications** of these Automata are given as follows:

**1.**[**Finite Automata (FA)**](https://www.geeksforgeeks.org/toc-finite-automata-introduction/)**–**

* For the designing of lexical analysis of a compiler.
* For recognizing the pattern using regular expressions.
* For the designing of the combination and sequential circuits using Mealy and Moore Machines.
* Used in text editors.
* For the implementation of spell checkers.

**2.**[**Push Down Automata (PDA)**](https://www.geeksforgeeks.org/theory-of-computation-pushdown-automata/)**–**

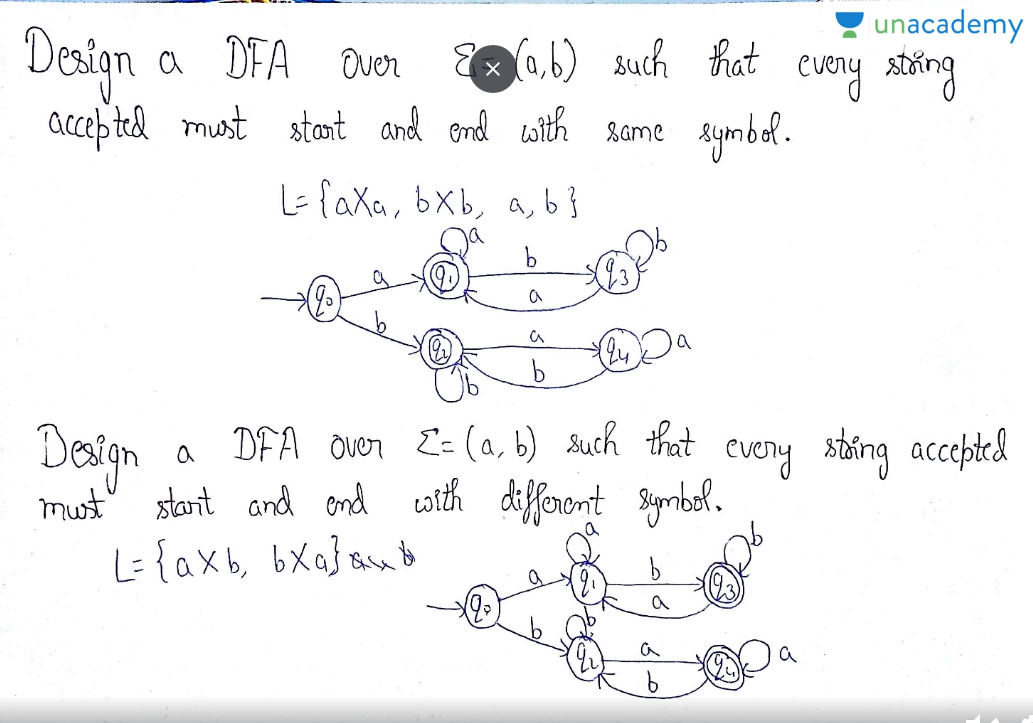
* For designing the parsing phase of a compiler (Syntax Analysis).
* For implementation of stack applications.
* For evaluating the arithmetic expressions.
* For solving the Tower of Hanoi Problem.

**3. Linear Bounded Automata (LBA) –**

* For implementation of genetic programming.
* For constructing syntactic parse trees for semantic analysis of the compiler.

**4.**[**Turing Machine (TM)**](https://www.geeksforgeeks.org/turing-machine/)**–**

* For solving any recursively enumerable problem.
* For understanding complexity theory.
* For implementation of neural networks.
* For implementation of Robotics Applications.
* For implementation of artificial intelligence.

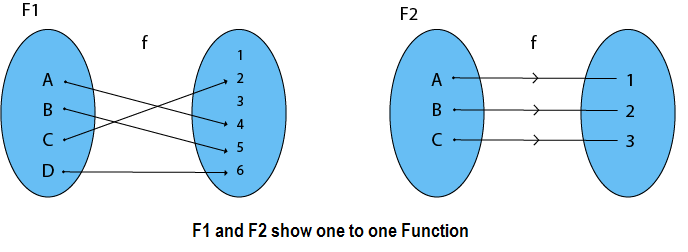


**Q2. What is the function also write the different type of functions?**

Ans. A **function** is something that associates each element of a set with an element of another set (which may or may not be the same as the first set). ... As you might have noticed, a **function** is quite like a relation . In fact, formally, we define a **function** as a special type of binary relation.

Types of Functions

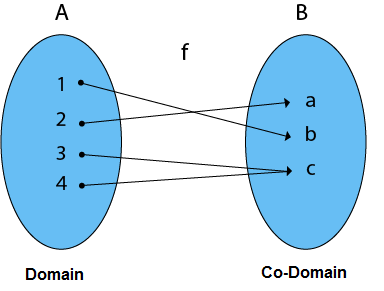
**1. Injective (One-to-One) Functions:** A function in which one element of Domain Set is connected to one element of Co-Domain Set.



**2. Surjective (Onto) Functions:** A function in which every element of Co-Domain Set has one pre-image.

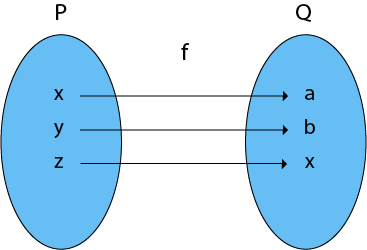
**Example:** Consider, A = {1, 2, 3, 4}, B = {a, b, c} and f = {(1, b), (2, a), (3, c), (4, c)}.

It is a Surjective Function, as every element of B is the image of some A



Note: In an Onto Function, Range is equal to Co-Domain.

**3. Bijective (One-to-One Onto) Functions:** A function which is both injective (one to - one) and surjective (onto) is called bijective (One-to-One Onto) Function.



**Example:**

1. Consider P = {x, y, z}
2. Q = {a, b, c}
3. and f: P → Q such that
4. f = {(x, a), (y, b), (z, c)}

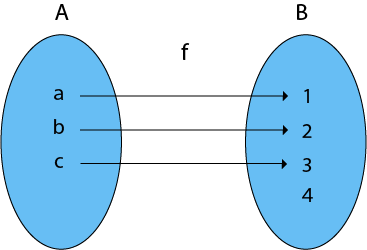
The f is a one-to-one function and also it is onto. So it is a bijective function.

**4. Into Functions:** A function in which there must be an element of co-domain Y does not have a pre-image in domain X.

**Example:**

1. Consider, A = {a, b, c}
2. B = {1, 2, 3, 4}   and f: A → B such that
3. f = {(a, 1), (b, 2), (c, 3)}
4. In the function f, the range i.e., {1, 2, 3} ≠ co-domain of Y i.e., {1, 2, 3, 4}

Therefore, it is an into function

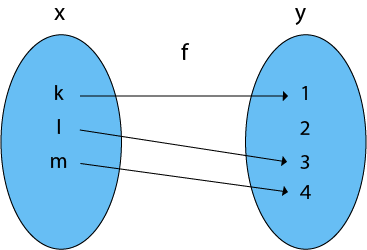


**5. One-One Into Functions:** Let f: X → Y. The function f is called one-one into function if different elements of X have different unique images of Y.

**Example:**

1. Consider, X = {k, l, m}
2. Y = {1, 2, 3, 4} and f: X → Y such that
3. f = {(k, 1), (l, 3), (m, 4)}

The function f is a one-one into function

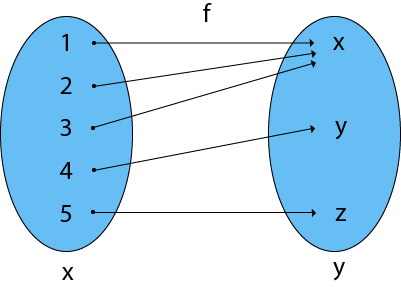


**6. Many-One Functions:** Let f: X → Y. The function f is said to be many-one functions if there exist two or more than two different elements in X having the same image in Y.

**Example:**

1. Consider X = {1, 2, 3, 4, 5}
2. Y = {x, y, z} and f: X → Y such that
3. f = {(1, x), (2, x), (3, x), (4, y), (5, z)}

The function f is a many-one function

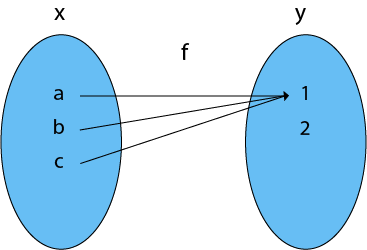


**7. Many-One Into Functions:** Let f: X → Y. The function f is called the many-one function if and only if is both many one and into function.

**Example:**

1. Consider X = {a, b, c}
2. Y = {1, 2} and f: X → Y such that
3. f = {(a, 1), (b, 1), (c, 1)}

As the function f is a many-one and into, so it is a many-one into function.

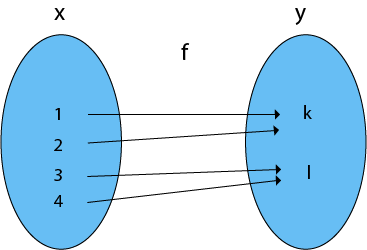


**8. Many-One Onto Functions:** Let f: X → Y. The function f is called many-one onto function if and only if is both many one and onto.

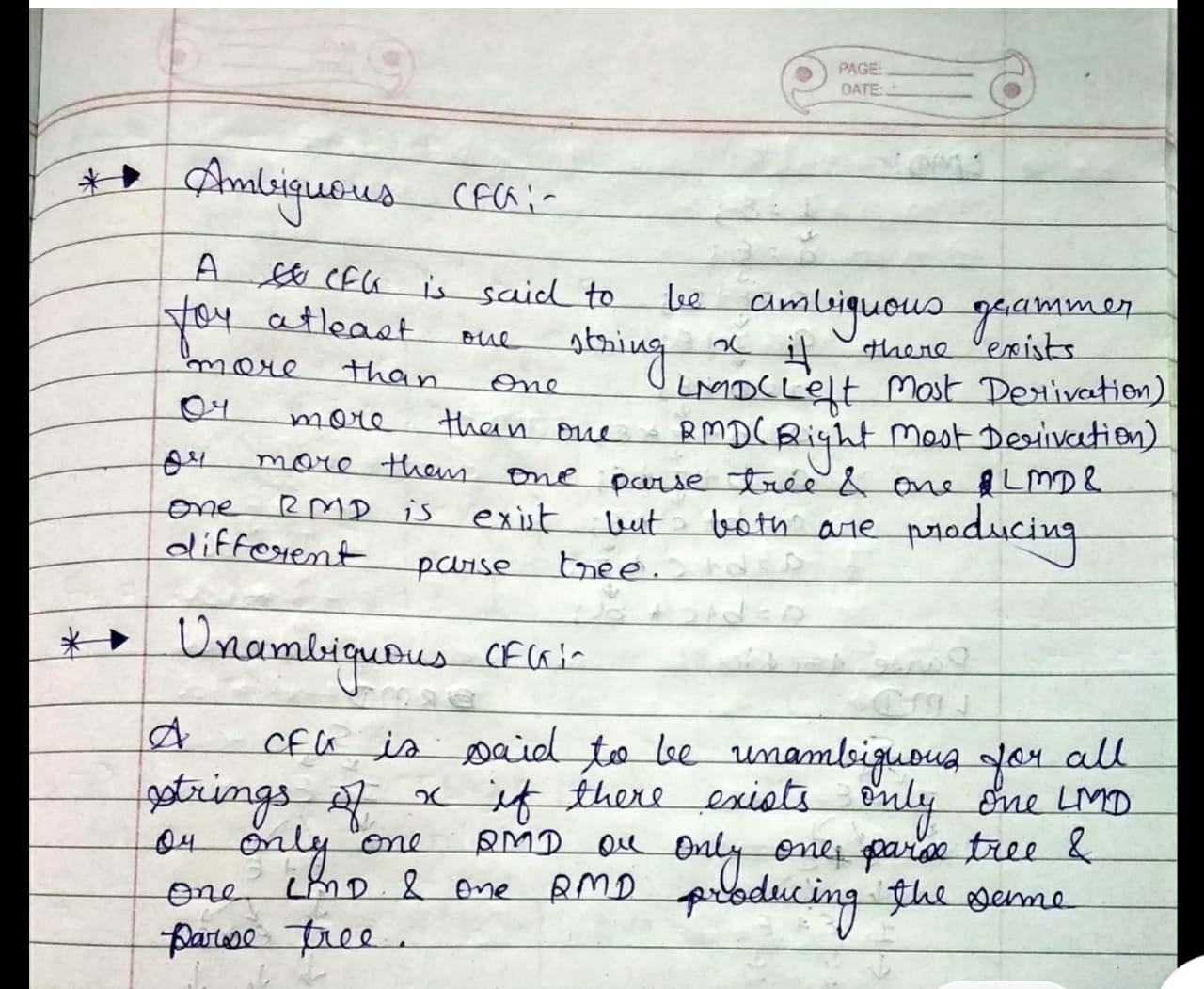
**Example:**

1. Consider X = {1, 2, 3, 4}
2. Y = {k, l} and f: X → Y such that
3. f = {(1, k), (2, k), (3, l), (4, l)}

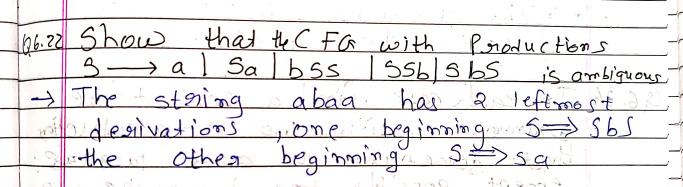
The function f is a many-one (as the two elements have the same image in Y) and it is onto (as every element of Y is the image of some element X). So, it is many-one onto function



**Ambiguous and non Ambiguous grammar**



**Show that the CFG with productions S —-» a|Sa|bSS|SSb|SbS is Ambiguous?**



\

**What is PDA**

Pushdown Automata is a finite automata with extra memory called

stack which helps Pushdown automata to recognize Context Free

Languages.  
   
A Pushdown Automata (PDA) can be defined as :

* Q is the set of states
* ∑is the set of input symbols
* Γ is the set of pushdown symbols (which can be pushed and popped from stack)
* q0 is the initial state
* Z is the initial pushdown symbol (which is initially present in stack)
* F is the set of final states
* δ is a transition function which maps Q x {Σ ∪ ∈} x Γ into Q x Γ\*. In a given state, PDA will read input symbol and stack symbol (top of the stack) and move to a new state and change the symbol of stack.

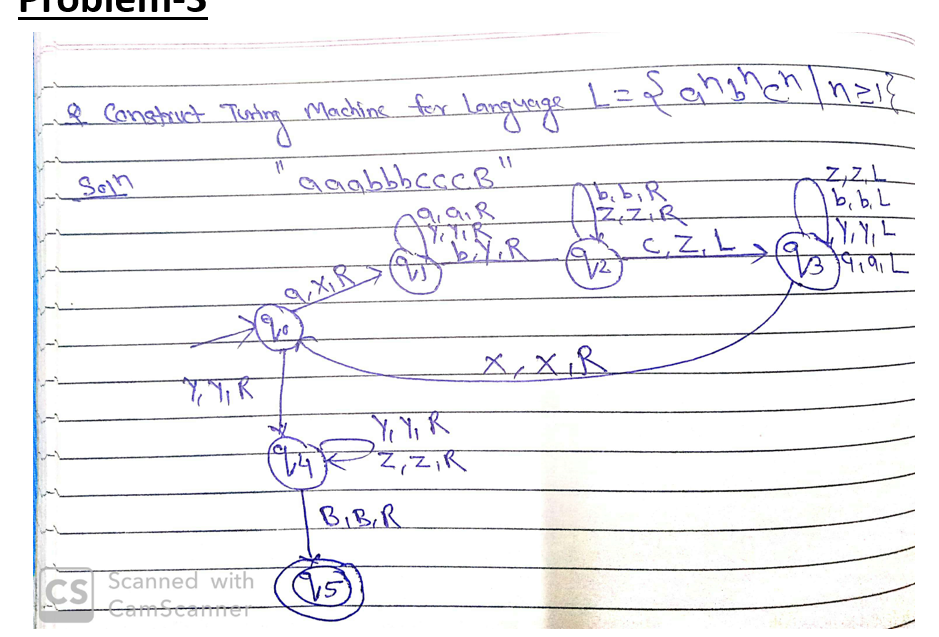
**What is turing Machine and its application**

Introduction: Turing Machine was invented by Alan Turing in 1936 and it is used to accept Recursive Enumerable Languages (generated by Type-0 Grammar)

A Turing machine consists of a tape of infinite length on which read and writes operation can be performed. The tape consists of infinite cells on which each cell either contains input symbol or a special symbol called blank. It also consists of a head pointer which points to cell currently being read and it can move in both directions.

**A TM is expressed as a 7-tuple (Q, T, B, ∑, δ, q0, F) where:**

* **Q** is a finite set of states
* **T** is the tape alphabet (symbols which can be written on Tape)
* **B** is blank symbol (every cell is filled with B except input alphabet initially)
* **∑** is the input alphabet (symbols which are part of input alphabet)
* **δ**is a transition function which maps Q × T → Q × T × {L,R}. Depending on its present state and present tape alphabet (pointed by head pointer), it will move to new state, change the tape symbol (may or may not) and move head pointer to either left or right.
* **q0** is the initial state
* **F** is the set of final states. If any state of F is reached, input string is accepted.



**Construct PDA for L = {WWR|W € (a+b)\*}**

For this problem Deterministic PDA is not possible because in DPDA , it can not able to differentiate W and WR .

So for this problem we can construct NPDA

