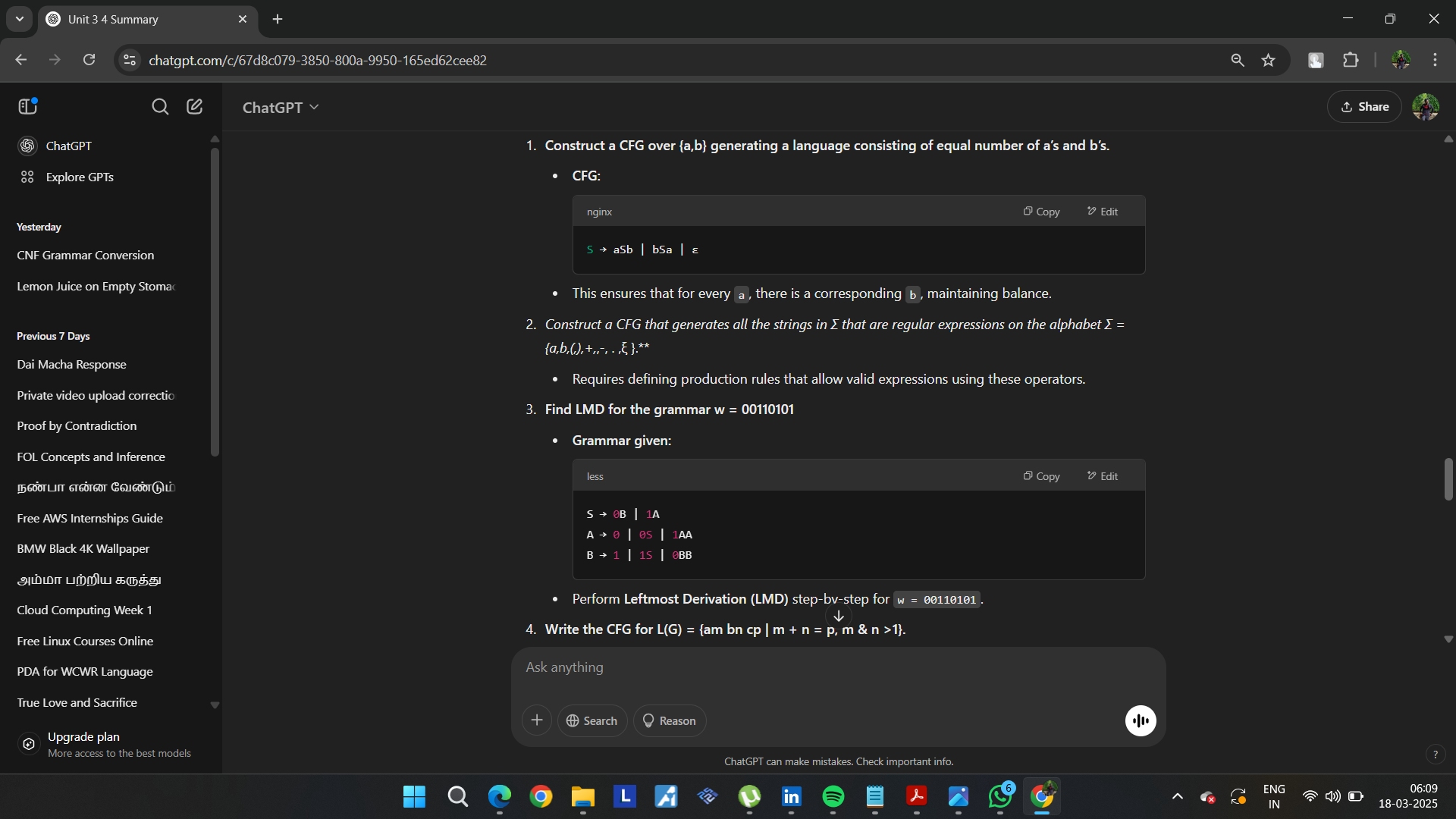
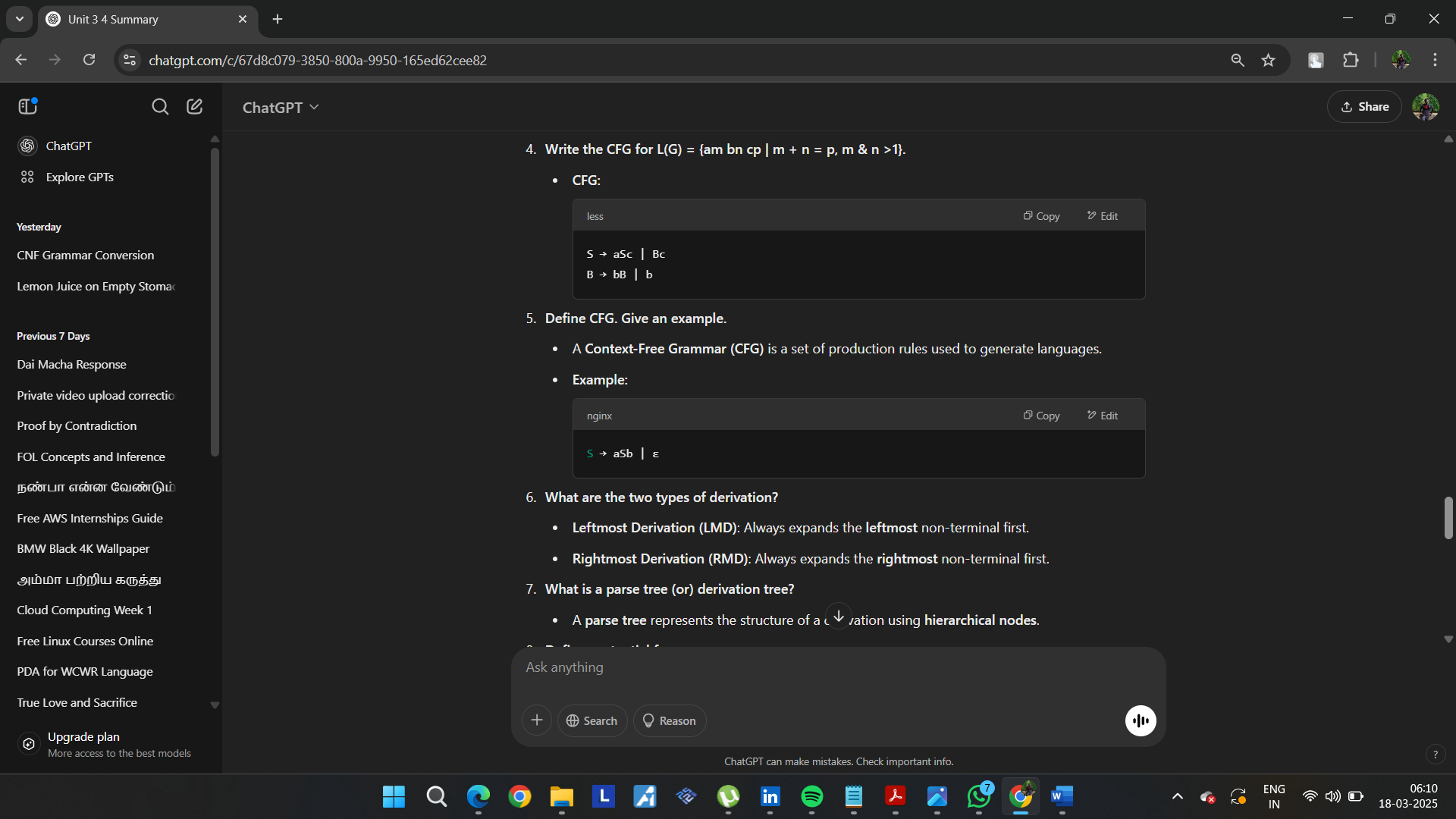
2 MARK

  
  
 **What are the two types of derivation?**

* **Leftmost Derivation (LMD)**: Always expands the **leftmost** non-terminal first.
* **Rightmost Derivation (RMD)**: Always expands the **rightmost** non-terminal first.

 **What is a parse tree (or) derivation tree?**

* A **parse tree** represents the structure of a derivation using **hierarchical nodes**.

 **Define sentential form.**

* A **sentential form** is any string derived from the start symbol using grammar rules.

 **Define Pushdown Automata (PDA).**

* A **PDA** is a computational model that extends finite automata with a **stack** to handle context-free languages.

 **Construct a PDA that accepts the language generated by the grammar**

S → aSbb

S → aab

δ(q, a, Z0) → (q, aZ0)

δ(q, a, a) → (q, aa)

δ(q, b, a) → (q, ε)

δ(q, b, a) → (q, ε)

**Construct a grammar for L = {an | n is odd}.**

* **CFG:**

S → aSa | a

A **language is accepted by final state** if the PDA reaches an accepting state after reading the input.  
  
  
**What are the different ways of language acceptance by a PDA?**

* **Final State Acceptance:** The PDA reaches a final state.
* **Empty Stack Acceptance:** The stack is **empty** at the end of input processing.

**Unit 3 – Context-Free Grammars (CFG)**

**15. Classify the grammar based on Noam Chomsky Hierarchy.**

* The **Noam Chomsky Hierarchy** classifies grammars into four types:
  1. **Type 0 (Unrestricted Grammar):** Can generate any recursively enumerable language.
  2. **Type 1 (Context-Sensitive Grammar - CSG):** Rules are of the form **α → β**, where |α| ≤ |β|.
  3. **Type 2 (Context-Free Grammar - CFG):** Rules have a single non-terminal on the left-hand side (**A → γ**).
  4. **Type 3 (Regular Grammar):** Rules are of the form **A → aB** or **A → a**, generating regular languages.

**16. Construct a CFG for the language L = { aⁿ bⁿ | n ≥ 0 }.**

* **CFG:**
* S → aSb | ε
* This ensures equal numbers of a and b in the correct order.

**Unit 4 – Pushdown Automata (PDA)**

**17. Draw PDA accepting the language L = { aⁿ c bⁿ | n > 0 }.**

* **Transitions:**
  1. Push a onto the stack.
  2. Read c without modifying the stack.
  3. Pop a when encountering b.
  4. Accept when the stack is empty.
* The PDA operates by pushing a on the stack for every input a, ignoring c, and popping for each b.

**18. What is an instantaneous description of PDA?**

* An **Instantaneous Description (ID)** of a PDA represents its **current state**, **remaining input**, and **stack content** in the form:
* (q, w, γ)
* Where:
  + q is the current state.
  + w is the remaining input.
  + γ is the current stack content.

**19. Define Deterministic PDA.**

* A **Deterministic PDA (DPDA)** is a PDA where:
  + **For each state and input symbol, at most one transition exists**.
  + **No ε-moves** are allowed (except under specific conditions).
* **DPDAs** cannot recognize all context-free languages, unlike **non-deterministic PDAs (NPDAs)**.

**20. Does a Pushdown Automata have memory? Justify.**

* Yes, a **PDA has memory** in the form of a **stack**.
* The **stack** allows a PDA to remember an **arbitrary number of symbols**, enabling it to recognize **context-free languages**.

**21. What is the additional feature PDA has when compared with NFA? Is PDA superior over NFA in the sense of language acceptance? Justify.**

* **Additional Feature:**
  + A **PDA** has a **stack**, which gives it the ability to recognize **context-free languages**.
  + An **NFA** (Non-deterministic Finite Automaton) does **not** have a stack and can recognize **only regular languages**.
* **Superiority:**
  + **Yes, PDA is superior to NFA** in terms of language acceptance because:
    - **NFA** can only accept **regular languages**.
    - **PDA** can accept **context-free languages**, which are more powerful than regular languages.

Let me know if you need explanations or diagrams for PDA constructions! 🚀

----------------------------------------------------------------------------------------------------------------------------------------------  
  
  
Here are the answers to the **Unit 4 (CO5) questions**:

**Context-Free Grammars and Normal Forms**

1. **Give the general forms of CNF.**
   * **Chomsky Normal Form (CNF)** requires that all production rules be of the form:
     1. **A → BC**, where **A, B, and C** are **non-terminals**, and **B & C are not start symbols**.
     2. **A → a**, where **a** is a **terminal**.
     3. **S → ε** (if the language contains an empty string).
2. **Name any four closure properties of regular languages.**
   * Regular languages are **closed under**:
     1. **Union** (L1 ∪ L2 is regular).
     2. **Concatenation** (L1L2 is regular).
     3. **Kleene Star** (L\* is regular).
     4. **Intersection** (L1 ∩ L2 is regular).
3. **State the two normal forms and give an example.**
   * **Chomsky Normal Form (CNF)**
   * S → AB | a
   * A → a | b
   * B → b
   * **Greibach Normal Form (GNF)**
   * S → aA | bB | a
   * A → a | bA
   * B → b
   * In **GNF**, the right-hand side **must start with a terminal**.
4. **Convert the given grammar into Greibach Normal Form (GNF):**
5. S → ABb | a
6. A → aaA | B
7. B → bAb
   * The conversion involves **eliminating left recursion** and ensuring that each rule starts with a **terminal**.
8. **Is L = {aⁿbⁿcⁿ | n ≥ 1} context-free? Justify.**
   * **No, L is not context-free.**
   * This language **requires a PDA to compare three different counts (a, b, c)**, which is **not possible** with a single stack.
   * **Proof:** **Pumping Lemma** for CFLs fails.
9. **Find L(G) for:**
10. S → aCa
11. C → aCa | ε
    * The language contains **palindromes with an equal number of 'a' on both sides**, e.g., {aa, aaaa, aaaaaa, ...}.
12. **Find the language generated by the given CFG:**
13. S → aSb | ab
    * The language consists of strings with **equal numbers of 'a' and 'b' in balanced pairs**:  
      {ab, aabb, aaabbb, ...}.
14. **Convert the given grammar into an equivalent one with no unit productions and no useless symbols.**
    * The conversion involves:
      + **Removing unit productions** (A → B).
      + **Eliminating unreachable or useless symbols**.
15. **Construct a CFG without ε-productions:**
16. S → a | Ab | aBa
17. A → b | ε
18. B → b | A
    * **After removing ε-productions:**
    * S → a | Ab | aB | aBa
    * A → b
    * B → b | b
19. **Convert the given grammar by removing unit productions and useless symbols.**
    * **Remove unit productions:**
    * S → CB
    * B → 1B | 1
    * C → 0C | 0
    * **Final equivalent grammar:**
    * S → CB
    * B → 1B | 1
    * C → 0C | 0

**Turing Machines (TM) and Automata**

1. **Compare Finite Automata (FA) and Turing Machine (TM).**
   * **FA:** Accepts **regular languages**.
   * **TM:** Accepts **recursively enumerable languages**, has **unlimited memory**, and can perform **computation**.
2. **Define GNF with an example.**
   * **Greibach Normal Form (GNF)**: Each production must start with a **terminal**.
   * A → aB
   * B → bC | ε
   * C → c
3. **What is a Turing Machine?**
   * A **Turing Machine (TM)** is a **mathematical model of computation** that manipulates symbols on an **infinite tape** according to a set of rules.
4. **Preliminary simplifications applied to grammars before conversion to normal forms:**
   * **Removing ε-productions**.
   * **Removing unit productions**.
   * **Eliminating useless symbols**.
5. **What is a multiple-track Turing machine?**
   * A **multiple-track TM** has **multiple rows** on its tape, allowing **parallel processing** of different data.
6. **Construct a TM for computing the 2’s complement of a binary number.**
   * **Steps:**
     + Scan from left to right, find the **first 1**, and invert the remaining bits.
     + Transition between states to perform this operation.
7. **List out different techniques for TM construction.**
   * **Storage in Tape**
   * **Multiple Tracks**
   * **Multiple Tapes**
   * **Non-Deterministic TM (NTM)**
   * **Universal TM (UTM)**
8. **What are the conventional notations of PDA?**
   * A **PDA** is represented as:
   * M = (Q, Σ, Γ, δ, q₀, Z₀, F)

Where:

* + - Q = Set of states
    - Σ = Input alphabet
    - Γ = Stack alphabet
    - δ = Transition function
    - q₀ = Start state
    - Z₀ = Initial stack symbol
    - F = Set of accepting states

1. **Design a TM that accepts the language of even integers written in binary.**
   * **Steps:**
     + Move right until the **last digit**.
     + Check if the **last digit is 0** (even number).
     + Accept if 0, reject if 1.
2. **Define Multi-Tape TM.**
   * A **multi-tape Turing Machine** has **multiple tapes** and **multiple heads**, each operating independently.
   * It can **increase computational efficiency** by allowing **parallel read/write operations**.

Let me know if you need explanations or diagrams! 🚀