INDIVIDUAL ASSSIGNMENT 01

COMPILER DESIGN

ABEL ANDUALEM ID 1411130

ASSIGNMENT 01

1. **(Theory):** Explain the difference between **top-down** and **bottom-up parsing**.
2. **(C++):** Write a C++ program to check if a string contains **balanced curly braces {}**.
3. **(Problem-solving):** Given grammar:
4. S →aSb | ε

   Draw the **parse tree** for "aabb".

1. Theory: Difference Between Top-Down and Bottom-Up Parsing

Top-Down Parsing (LL):

- **Direction:** Starts from start symbol (root) → builds downward to leaves
- **Approach:** Uses leftmost derivation, tries to predict which production to apply
- **Examples:** Recursive Descent, LL(1), Predictive Parsing
- **Advantages:** Easier to understand and implement manually
- **Limitations:** Cannot handle left recursion directly
- **Analogy:** Building a tree starting from the trunk

Bottom-Up Parsing (LR):

- **Direction:** Starts from input tokens (leaves) → builds upward to root
- **Approach:** Uses rightmost derivation in reverse, reduces input to start symbol
- **Examples:** SLR, LR(1), LALR, Operator-Precedence
- **Advantages:** More powerful, handles left recursion
- **Limitations:** More complex to implement manually
- **Analogy:** Building a tree starting from leaves and twigs

---

2. C++: Balanced Curly Braces Checker

```cpp
#include <iostream>
#include <stack>
using namespace std;

bool isBalanced(string s) {
    stack<char> st;
    for(char c : s) {
        if(c == '{') st.push(c);
        else if(c == '}') {
            if(st.empty()) return false;
            st.pop();
        }
    }
    return st.empty();
}
```

3. Parse Tree for "aabb"

Grammar: S → aSb | ε

Parse Tree:

```text
        S
       /|\
      a S b
       /|\
      a S b
         |
         ε
```

Derivation: S → aSb → aaSbb → aabb