Lab2B-Deliverables

Submission Format

- Students must submit a **single compressed archive** named: lab2b_rollno.tar.gz
- **Do not** submit compiled binaries (e.g., `a.out`, `lex.yy.c`, `y.tab.c`, `.o` files). Only submit source code and documentation.

Expected Directory Structure

```
lab2b_rollno/
├── src/
│  ├── lexer.l
│  ├── parser.y
│  ├── ast.c / ast.h (or equivalent)
│  └── main.c
├── Makefile
├── tests/
│  ├── valid/
│  └── invalid/
├── report.pdf
└── README.md
```

Students may deviate from this structure **only if clearly documented** in `README.md`.

---

1. Source Code

- Lexical analysis must be implemented using **Flex**.
- Parsing must be implemented using **Bison**.
- AST construction is **mandatory**.
- Grammar must support:
  - variable declaration and assignment
  - if / else
  - while loops
  - arithmetic and comparison expressions
- Code must compile and run on the lab machines.

---

2. Makefile

- `make` should build the parser executable.
- `make clean` should remove all generated files.
- No manual invocation of Flex/Bison should be required.

---

3. Tests

- Students must include **at least 10 test programs**, divided into:
  - Valid programs (expected to parse successfully)
  - Invalid programs (expected to fail gracefully)
- Tests should cover all supported language constructs.

> **Important:**
> During evaluation, TAs will run the submitted parser on **additional test cases**, including more complex and nested programs.
> Submissions must not rely on hardcoded assumptions or limited test coverage.

---

4. Technical Report (3–4 pages)

The report should include:

- Description of the designed language
- Lexer rules and tokenization strategy
- Grammar design and operator precedence
- AST structure and construction
- Error handling strategy
- Limitations and possible extensions

---

5. Demo and Evaluation

- A **live demo** will be conducted by the TAs.
- Students must:
    - Run their code on test cases provided by the TAs
    - Explain the grammar, lexer rules, and AST construction
- Use of AI tools is permitted, but students must be able to:
    - Clearly explain their code and design decisions
    - Answer questions about grammar rules and parsing behavior
      Failure to explain the implementation may result in loss of marks.

---

6. README [Important]

Must include:

- Build instructions
- How to run the parser
- How to run tests
- Any assumptions or deviations from the sample grammar.

7. Optional Extensions (Not Required)

- Script files (parse from file instead of stdin)
- Error recovery
- Functions or additional constructs
- Pretty-printing AST