# UNIVERSITY OF TECHNOLOGY, JAMAICA

**SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY**

COMPUTER SCIENCE MAJOR

THEORY OF COMPUTATION (CIT3006)

GROUP PROJECT (20%)

Documentation Due Date: April 4, 2026 Interview: Week 13

## Syntax Checker for a Simple Programming Language

In a group of 3-4 students design and implement a syntax checker for a small, well-defined programming language using Context-Free Grammars (CFGs) and parsing techniques. The goal is to demonstrate your group's ability to:

- Construct grammars
- Analyse ambiguity
- Apply grammar transformations
- Implement a working parser
- Validate input programs
- Produce derivations or parse trees

This project mirrors real-world tasks performed in compiler design, interpreters, IDEs, and markup validators.

Each group must choose one of the following language domains:

1. Arithmetic Expressions with Precedence
2. Conditional Statements (if–then–else)
3. HTML-like Nested Tags
4. Simple Assignment Statements

Your syntax checker must determine whether an input program/document written in your chosen language is syntactically valid.

Use of Libraries and Tools:

Students may use standard programming libraries for data structures, string manipulation, and input/output. However, automatic parser generators or grammar-processing tools (e.g., ANTLR, YACC, Bison) are not permitted, as the objective of this assignment is to demonstrate an understanding of grammar design, parsing logic, and theoretical principles.

Any external libraries used must be clearly documented and justified in the report.

**Core Theoretical Requirements**
Your submission must demonstrate mastery of the following Theory of Computation concepts:

1. Context-Free Grammar Design
   - Define terminals and non-terminals
   - Provide a complete set of production rules
   - Explain the structure and rationale of your grammar
2. Ambiguity Analysis
   - Identify whether your grammar is ambiguous
   - Provide examples illustrating ambiguity (if present)
   - Discuss how ambiguity affects parsing
3. Parsing - Implement one of the following:
   - Top-down parser (e.g., recursive descent)
   - Bottom-up parser (e.g., shift-reduce, CYK, etc.)

   Your parser must:
   - Accept syntactically valid inputs
   - Reject invalid inputs
   - Produce parse trees or leftmost/rightmost derivations
   - Provide meaningful syntax error messages
4. Grammar Normalization
   - Convert your grammar to Chomsky Normal Form (CNF)
   - Show each transformation step
   - Explain why CNF is useful in parsing

**Functional Requirements**
Your syntax checker must:

- Read an input program/document
- Validate it using your grammar
- Generate a parse tree or derivation sequence
- Detect and report syntax errors
- Demonstrate correctness using test cases

**Deliverables**
This is a group-based assignment to be completed in teams of 3-4 students. Group formation must be finalized by the end of Week 3. No individual submissions will be accepted. Each group must submit the following:

1. Grammar Specification (Terminals, Non-terminals, Production rules, Start symbol, Ambiguity discussion)
2. Parser Implementation: submit source code (any language) along with clear instructions for running the parser.
3. Test Suite
   - Accepted examples (valid programs)
   - Rejected examples with explanations
   - Screenshots or logs of parser output

4. A written report that includes
   (i) grammar design and justification
   (ii) ambiguity analysis
   (iii) parsing approach and algorithm
   (iv) CNF conversion steps
   (v) parse trees or derivations for sample inputs
   (vi) discussion of correctness and limitations
5. Interview & Demonstration
   During Week 13, groups will need to explain their grammar, walk through parsing logic, demonstrate the syntax checker and answer questions on theory and implementation.

**Assessment Criteria**

Marks will be awarded based on:

| Component | Weight |
|---|---|
| Grammar accuracy and completeness | 25% |
| Parser correctness and robustness | 30% |
| Theoretical justification (ambiguity, CNF, parsing) | 25% |
| Quality of report and test cases | 15% |
| Interview/presentation | 5% |

Students must adhere to the UTech, Ja. academic regulations, including those governing plagiarism and academic misconduct (see Regulation 5). All work must be original and properly referenced.

<u>Late Submission</u>

- Submissions received within 24 hours after the deadline will incur a 10% penalty on the total mark awarded.
- Submissions received between 24 and 48 hours after the deadline will incur a 20% penalty on the total mark awarded.
- Submissions received more than 48 hours after the deadline will receive a mark of zero but will be recorded as submitted.
- Late penalties apply to both individual and group components.

## Assessment Rubric

Total: 20% of course grade (100 marks)

---

**Grammar Design (25 marks)**

1. Grammar Specification (10 marks)
   - Clear definition of terminals and non-terminals (3)
   - Well-structured production rules (4)
   - Correct start symbol and language coverage (3)
2. Grammar Quality & Rationale (10 marks)
   - Grammar accurately models the chosen language (5)
   - Explanation of design choices (5)
3. Ambiguity Analysis (5 marks)
   - Correct identification of ambiguity OR justification of unambiguity (3)
   - Examples illustrating ambiguity or proof of non-ambiguity (2)

---

**Parsing Implementation (30 marks)**

1. Parser Correctness (15 marks)
   - Correct acceptance of valid inputs (5)
   - Correct rejection of invalid inputs (5)
   - Meaningful syntax error messages (5)
2. Parsing Method & Logic (10 marks)
   - Correct implementation of top-down or bottom-up parsing (5)
   - Clear explanation of algorithmic approach (5)
3. Parse Trees / Derivations (5 marks)
   - Accurate parse trees or leftmost/rightmost derivations (5)

---

**Theoretical Justification (25 marks)**

1. CNF Conversion (15 marks)
   - Explanation of CNF's utility in parsing (3)
   - Correct step-by-step transformation (7)
   - Explanation of each transformation (5)
2. Discussion of Grammar Properties (10 marks)
   - Clarity on grammar structure, recursion, precedence, associativity, etc. (5)
   - Discussion of limitations or design trade-offs (5)

---

**Report Quality & Test Suite (15 marks)**

1. Report Structure & Clarity (5 marks)
   - Logical organization, readability, academic tone (5)

2. Test Cases (10 marks)
    - Valid examples with parse trees/derivations (5)
    - Invalid examples with explanations (5)

---

**Interview & Demonstration (5 marks)**

- Clear explanation of grammar and parser (2)
- Ability to answer theoretical questions (2)
- Professionalism and clarity of demonstration (1)

---