***Name: Kamil Roginski  
Project: CMSC 315 Programming Project 3  
Date: April 23, 2025***

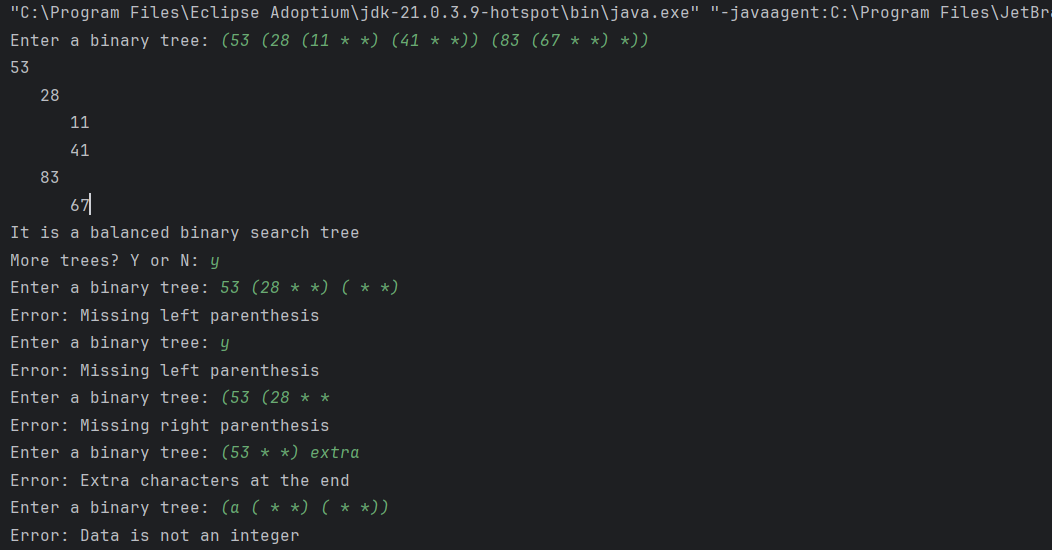
**Lessons Learned**

Designing this project reinforced how critical a solid, immutable data‐structure foundation is for building reliable software. By parsing a nested string representation, I honed my attention to detail around syntax and error handling—important for any production‐grade parser. Constructing balanced trees from sorted data illustrated how powerful divide‐and‐conquer patterns can be. As a professional engineer with an entrepreneurial mindset, I appreciated the value of clean separation of concerns: parsing, core logic, I/O, and exception types each live in their own class. Finally, writing a thorough test plan and clear UML diagram reminded me that good documentation accelerates collaboration and maintenance—key lessons for any business or startup I might found in the future.

**Test Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Outcome** | **Actual Outcome** |
| **Test Case 1 (Balanced BST)** | (53 (28 (11 \* \*) (41 \* \*)) (83 (67 \* \*) \*)) | Prints indented tree; message: “It is a balanced binary search tree” | Prints indented tree; message: “It is a balanced binary search tree” |
| **Test Case 2 (Missing Left Parenthesis)** | 53 (28 \* \*) ( \* \*) | Error: “Missing left parenthesis” | Syntax: missing leading ( |
| **Test Case 3 (Missing Right Parenthesis)** | (53 (28 \* \* | Error: “Missing right parenthesis” | Error: “Missing right parenthesis” |
| **Test Case 4 (Extra Characters at End)** | (53 \* \*) extra | Error: “Extra characters at the end” | Error: “Extra characters at the end” |
| **Test Case 5 (Data Not Integer)** | (a ( \* \*) ( \* \*)) | Error: “Data is not an integer” | Error: “Data is not an integer” |

**Test Case 1-5:**



**UML Diagram**

A screenshot of a computer program

AI-generated content may be incorrect.