8 Lecture - CS301

Important Subjective

1. **What is the main advantage of postfix notation over infix notation?**
   Answer: Postfix notation eliminates the need for parentheses to indicate the order of operations.
2. **What is the role of a stack in converting an infix expression to postfix notation?**
   Answer: The stack is used to keep track of operators and their precedence levels.
3. **How do you handle errors while converting an infix expression to postfix notation?**
   Answer: By checking for balanced parentheses and errors during scanning.
4. **What is the first step in converting an infix expression to postfix notation?**
   Answer: Initializing an empty stack and postfix expression.
5. **How do you handle operators with equal precedence levels while converting infix to postfix notation?**
   Answer: Operators with equal precedence levels are added to the postfix expression based on the associativity rules (left to right or right to left).
6. **What happens to a left parenthesis when converting infix to postfix notation?**
   Answer: The left parenthesis is pushed onto the stack.
7. **What is the final step in converting an infix expression to postfix notation?**
   Answer: Pop any remaining operators off the stack and add them to the postfix expression.
8. **What is the difference between infix notation and postfix notation?**
   Answer: In infix notation, operators are written between their operands, while in postfix notation, operators are written after their operands.
9. **What are the advantages of using a postfix notation over infix notation?**
   Answer: Postfix notation eliminates the need for parentheses to indicate the order of operations, and it is easier to evaluate expressions using a stack-based algorithm.
10. **How can you convert an infix expression with nested parentheses to postfix notation?**
    Answer: By using a stack-based algorithm to remove the nested parentheses and convert the expression to postfix notation.