**Project Proposal: Infix to Postfix Calculator Using Stack**

**1. Project Title:**

Infix to Postfix Calculator Using Stack

**2. Introduction:**

Mathematical expressions are commonly written in **infix notation** (e.g., A + B \* C), but computers and compilers often prefer **postfix notation** (e.g., A B C \* +). Converting infix expressions to postfix simplifies evaluation and avoids ambiguity. This project proposes a calculator that converts infix expressions to postfix and evaluates them using a **stack-based approach**.

**3. Problem Statement:**

Manual conversion of infix expressions to postfix is tedious and error-prone. Developing an automated calculator will help users quickly convert and evaluate expressions without ambiguity, making it useful for students, programmers, and software applications.

**4. Objectives:**

* Implement an algorithm to convert infix expressions to postfix using a stack.
* Develop an evaluation mechanism to compute results from postfix expressions.
* Provide a user-friendly interface for entering expressions and displaying results.
* Ensure efficient handling of operator precedence and associativity.

**5. Scope of the Project:**

* Input validation for mathematical expressions.
* Implementation of **Dijkstra’s Shunting Yard Algorithm** for conversion.
* Stack-based postfix evaluation.
* GUI or command-line interface for usability.
* Error handling for invalid expressions.

**6. Technology Stack:**

* **Programming Language:** Python/C++/Java
* **Data Structure:** Stack (implemented using arrays or linked lists)
* **Frontend (if GUI is included):** Tkinter (Python) or Java Swing
* **Backend (if applicable):** Flask (Python)

**7. Methodology:**

1. **Infix to Postfix Conversion:**
   * Use a stack to handle operators based on precedence and associativity.
2. **Postfix Expression Evaluation:**
   * Use a stack to evaluate the postfix expression efficiently.
3. **User Interface Development:**
   * Implement a simple GUI or CLI for input and output.
4. **Testing & Optimization:**
   * Handle edge cases (e.g., parentheses, division by zero).
5. **Final Deployment:**
   * Package the program for end-user accessibility.

**8. Expected Outcomes:**

* A functional calculator that converts infix expressions to postfix.
* Accurate and efficient evaluation of postfix expressions.
* A simple and intuitive user interface for ease of use.

**9. Timeline:**

| **Task** | **Duration** |
| --- | --- |
| Algorithm Implementation | 2 weeks |
| Infix to Postfix Conversion | 2 weeks |
| Postfix Evaluation Implementation | 2 weeks |
| User Interface Development | 2 weeks |
| Testing and Debugging | 2 weeks |
| Final Documentation & Deployment | 1 week |

**10. Budget Estimation:**

* Development Tools: Free (Open-source languages & libraries)
* Hardware (if needed): $100
* Miscellaneous: $50 **Total Estimated Cost:** ~$150

**11. Conclusion:**

This project provides an efficient way to convert and evaluate mathematical expressions using a **stack-based approach**. By automating the conversion process, it enhances accuracy and usability for students, programmers, and mathematical applications.