Assignment 2:

- Pratyush Saini (2019CS10444)
- Anirudha Kulkarni (2019CS50421)

Preface:

Postfix expressions are a way of representing mathematical expression consisting of operators and operands wherein, the position of operators leads the position of operands. The following MIPS assembly program evaluates the value of an input postfix expression by means of a stack data structure.

Input:

The program takes in a single line of input, essentially an input string consisting of digits[0-9] and mathematical operators [+-*x] (except division). The default maximum length of string is set to 100 and can be changed as per specifications

Algorithm / Approach:

The algorithm used for evaluating the input postfix expression is based on Stack Data Structure.

1. The input string is parsed in left to right fashion reading a single character at a time.
2. **Case 1**- If the character is some digit [0-9], then the character is pushed into top of the stack and the stack pointer is incremented by a single unit offset.
3. **Case 2** - If the character is some operand [+-*x] (conditioned by the ascii value of the operand), then top two elements of the stack are popped and the stack

pointer is decremented by two unit offset. The result after operating the operator on the two popped elements is pushed to the top of the stack. The stack pointer is incremented by a single unit offset.
4. Finally, the stack will contain a single element, the final value of the postfix expression.

Register's used:

- $s0 - for length of input string
- $s1 - address of input string
- $t3 - first pop value
- $t4 - second pop value
- $sp - top pointer of the stack

Error Handling / Faulty Test Cases:

- **Invariant:** In case an operand strikes when parsing the input string, the size of stack should be at least 2.
- To handle such error, the size of stack is stored in register $s2. After each push or pop operation, the size variable of stack is incremented/ decremented accordingly. The size value is validated whenever some operator is encountered and the appropriate error is thrown before terminating the program.
- **Invariant:** The size of stack after parsing the whole string should be equal to one, failing which the error for Invalid Input string is raised.
- In case the character read is not in the input domain of the string, Invalid Input expression error is raised.

Output

The output consists of:

- State of program at each occurence of an operator. That is, whenever an operator is read, the operation executed at that particular iteration is printed to the console.
- The final value of the input expression is displayed at the end of console, in case of valid input postfix expression. Else, appropriate error is raised.

Test Cases:

Find extensive testing details under Testing/README.md

1. simple string with all 4 operations
2. string with any character other than integers and operators +-*

3. Empty expression
4. Case of faulty Input string where the stack becomes empty in the middle of parsing
5. input string where the stack's size is not 1 at the end of parsing: