Ruby Calculator – Specification Document

1. Purpose

A simple command-line calculator in Ruby that supports basic arithmetic operations with integer operands, handles negative numbers, and provides integer or floating-point division.

2. Supported Operations

Operation	Symbol	Behavior
Addition	+	Integer addition of two operands
Subtraction	-	Integer subtraction of two operands
Multiplication	*	Integer multiplication of two operands
Integer Division	/	Divides first operand by second operand, result truncated to integer
Floating-Point Division	//	Divides first operand by second operand, result as floating-point

3. Input Syntax

The calculator accepts single binary operations in the following format:

<operand1> <operator> <operand2>

3.1 Operands (operand1 and operand2)

- Integer numbers (positive or negative)
- Range: -2,147,483,648 to 2,147,483,647 (Ruby Integer limits)

3.2 Operator (operator)

- One of +, -, *, /, //

3.3 Spacing

- Optional spaces allowed around operands and operator.
- Valid input examples:
  - 6+4
  - 6 + 4
  - -5*-3
  - 12 // -4

3.4 Invalid input examples

- Multiple operations (e.g., 6+4-2)
- Non-numeric operands (e.g., a + b)
- Unsupported operators (e.g., %)

4. Error Handling

The calculator provides clear error messages for invalid input or operations:

4.1 Invalid Expression
- Returned if the input does not match the expected syntax.
- Example: 6 + 4 - 2, ++5

4.2 Division by Zero is not Possible
- Returned if the second operand is 0 when using / (integer division) or // (floating-point division).
- Example: 7 / 0, 10 // 0

4.3 Operands must be Integers
- Returned if any of the operand is not an integer.
- Example: a + 3, c / e

4.4 Unknown operator. Please enter a valid operator
- Returned if the operator is not one of the supported symbols (+, -, *, /, //).
- Example: 5 % 2, 4 ^ 2

5. Command Loop

- Users can repeatedly enter operations until they exit.
- Exit commands: exit, quit, q (case-insensitive)
- Behavior on exit: prints "Goodbye!" and terminates the program

6. Assumptions and Constraints

- Only integer operands are supported. Floating-point input is not accepted.
- The calculator handles one operation at a time.
- User input is assumed to be a single-line string per calculation.
- Division results adhere to integer truncation (/) or floating-point division (//).

7. Implementation

1. Print a welcome message and instructions for supported operations (+, -, *, /, //) and exit commands (exit, quit, q).
2. Enter a loop to continuously read user input, trim whitespace, and convert it to a string. Exit if the input is an exit command.
3. Attempt strict parsing using a regular expression to match two integers and a single operator. If matched, extract operands and operator.
4. If strict parsing fails, use a fallback regex to split the input into operand1, operator, and operand2. Return "Invalid Expression" if splitting fails.

5. Check operands for extra operator symbols and convert them to integers.
   Return "Operands must be integers" if conversion fails.
6. Validate the operator against the supported list. Return "Unknown operator. Please enter a valid operator!" if invalid.
7. Pass operands and operator to a helper function to perform the calculation, handling division by zero and floating-point division (//).
8. Print the result or error message and repeat the loop until the user exits.
9. On exit, print "Goodbye!".

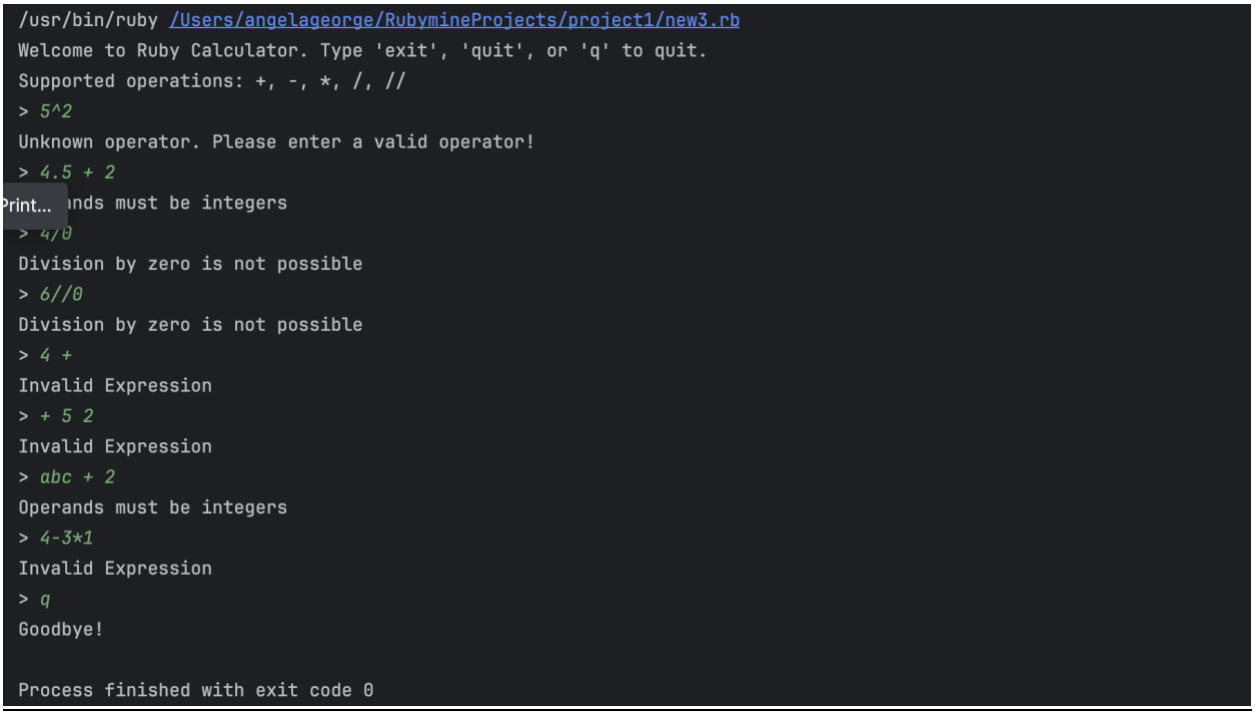
8. Output

8.1 Valid Cases

```
/usr/bin/ruby /Users/angelageorge/RubymineProjects/project1/new3.rb
Welcome to Ruby Calculator. Type 'exit', 'quit', or 'q' to quit.
Supported operations: +, -, *, /, //
> 5+2
7
> 5  - 3
2
> -4+4
0
> 3*5
15
> 10/2
5
> 10//3
3.3333333333333335
> 12345-54321
-41976
> -7*-8
56
> -128/4
-32
> exit
Goodbye!

Process finished with exit code 0
```

8.2 Invalid Cases



9. Limitations and Future Enhancements

- The current calculator only supports binary operations.
- No support for parentheses or operator precedence.
- Floating-point operands are not supported.
- Future work could include: multiple operations, more operators (%, **), and handling decimals.