Calculator

Description

The objective of this assignment is to create a calculator than can perform addition, subtraction, multiplication, and division. The user will also be able to enter in special characters to either clear the calculator's buffer or exit the calculator program.

The program should initially accept 2 operands and a mathematical operator (+, -, *, /) then store the results. The succeeding steps should only ask for the operator and a new input number. The operator would then be applied to the previously stored result (the buffer contains the result) and the new input number. The two special operators will be "c" and "x", where c will clear the buffer and x will exit the program.

The calculator will be expected to catch (i.e., recover from an error without crashing) division by zero and unknown operator errors, as shown in the examples below. If either error occurs, the buffer's value will remain the same.

Lastly, the running of the program should be continuous by default. In other words, the user should be able to perform as many calculations as they desire until they decide to exit the program. Note that this is different from the time difference assignment, where it was fine to let the program end after each calculation.

Design and Implementation

The program first needs to gather input from the user before it can proceed. Think carefully about what the type should be for the buffer, given the different types of numeric input (integer and floating-point).

As far as assumptions about the input, it is safe to assume that the user will always pass in a valid numeric entry for the 1st and 2nd inputs. In other words, your program does not need to catch bad input in the form of a string or char.

It is not safe to assume however that the first calculation will always be a valid one; the user could clear the buffer or exit the program at the first opportunity. Also, the user might accidentally divide by zero or enter an unknown operator during the first calculation. Your program should be able to handle these cases.

As for the standard operations of the calculator, make use of what you have learned about control flow structures. Be careful not to mix up operands with the buffer and visa versa; otherwise, your calculator may perform calculations incorrectly.

Example Output

Normal run

```
1st input: 3
op: +
2nd input: 2.5
ans: 5.5
op: *
more input: 3
ans: 16.5
op: c
ans: 0.0
op: +
more input: 10
ans: 10.0
op: x
```

Division by zero error

```
1st input: -3
op: *
2nd input: 2
ans: -6.0
op: /
more input: 0
Error: division by zero
op: +
more input: 10
ans: 4.0
```

Unknown operator error

```
1st input: -3
op: $
2nd input: 2
Error: Unknown operator $
op: +
more input: 6
ans: 3.0
```

Formatting and Style

It will be important to create descriptive variable names that show the purpose for each variable used - especially when it comes to the control flow statements. On the subject of control flow, make sure to exercise good style when it comes to writing out the different logical blocks (e.g., use indentations and place the curly brackets in sensible locations).

From the user's point of view, the messages printed out to the screen should be brief and clear. Also, as with the last assignment, make sure to clearly differentiate between lines with input and lines with output. It would be a good idea to print a message for when the buffer is cleared and when the program is about to exit.

Grading Rubric

Points	Criteria
1	Parses user input correctly
2	Performs calculations by case (+,-,*,/)
2	Handles special commands (clear buffer, quit)
2	Reasonable control flow and logic that allow for seamless execution of the program
2	Implements error handling (division by zero, unknown operator)
1	Good style demonstrated in the code; sensible formatting