Problem 4. Explain how edge cases are handled. Edge cases to address:

- ➢ Empty postfix expressions
- ➢ Malformed postfix expressions (insufficient operands, too many operands)
- ➢ Division by zero
- ➢ Invalid tokens (non-numeric operands, unsupported operators)
- ➢ Very large numbers or results
- ➢ Negative numbers in the expression

Didn't use any external source for this answer.

- The code checks the empty input list and raises a value error exception in Python.

For example it checks it with code " if not elts:" in evaluayePostfix function.

- The code is checking for malformed expressions, such as not enough operands or too many operands in constructBinaryTree and evaluatePostFix functions. For example, after an operation (+,-,*,/), it checks the stack size is at least 2.

It also checks if size > 1 after processing to detect to many operands.

- Divide by zero is checked if the evaluatePostFix function tries to do a division and the right node of the subtree is zero. It raises a dividebyzero error and it is caught by the main test function successfully.

- Invalid tokens are checked if the operators are not one of the operations (*+,-,*,/) or an invalid value (non-integer string) when it converts the string to integer using code num= int(elt) and it raises a ValueError exception.

- Python handles very large numbers with int() but it also depends on the computer the algorithm is running. If the computer is 32 bit or 64 bit etc. and also depends on the programming language used (C needs special handling where Python may be more flexible). I could define INTMAX = 2**31 – 1 and INTMIN = -2**31 for a 32 bit

machine for example and check if the values are bigger than max or smaller than the minimum values the computer/compiler/interpreter can handle.

- Negative numbers are handled again using the integer conversion with code num= int(elt) line. I also tested with additional expression added to the test file, for example: 3 4 + -2 * where a negative 2 is included in the test and successfully evaluated as -14.

Expected Output:

RUNNING TEST CASES FOR PROBLEM 1

P1 Test 1 passed

P1 Test 2 passed

P1 Test 3 passed

P1 Test 4 passed

P1 Test 5 passed

P1 Test 6 passed

P1 Test 7 passed

P1 Test 8 passed

P1 Test 9 passed

P1 Test 10 passed

P1 Test 11 passed

P1 Test 12 passed

P1 Test 13 passed

P1 Test 14 passed

P1 Test 15 passed

P1 Test 16 passed

P1 Test 17 passed

P1 Test 18 passed

P1 Test 19 passed

P1 Test 20 passed

P1 Test 21 passed

RUNNING TEST CASES FOR PROBLEM 2

P2 Test 1 passed

P2 Test 2 passed

P2 Test 3 passed

P2 Test 4 passed

P2 Test 5 passed

P2 Test 6 passed

P2 Test 7 passed

P2 Test 8 passed

P2 Test 9 passed

P2 Test 10 passed

P2 Test 11 passed

P2 Test 12 passed

P2 Test 13 passed

P2 Test 14 passed

P2 Test 15 passed

P2 Test 16 passed

P2 Test 17 passed

P2 Test 18 passed

P2 Test 19 passed

P2 Test 20 passed

P2 Test 21 passed

P2 Test 22 passed

RUNNING TEST CASES FOR PROBLEM 3

Test case 1 passed

Test case 2 passed

Test case 3 passed

Test case 4 passed

Test case 5 passed

Test case 6 passed

Test case 7 passed

Test case 8 passed

Test case 9 passed

Test case 10 passed

Test case 11 passed

Test case 12 passed

Test case 13 passed

Test case 14 passed

Test case 15 passed

Test case 16 passed

Test case 17 passed

Test case 18 passed

Test case 19 passed

Test case 20 passed

Test case 21 passed (division by zero handled)

Test case 22 passed (division by zero handled)