Name: Ammar Mohamed Abbas
ID :20220215
Section:S8
TA: Khaled Ahmed

Name: Begab Abdelghafar Abdelhady
ID: 20220521
Section : S8
TA : Khaled Ahmed

- The Algorithm -

Addition Algorithm (+):

Algorithm BigDecimalInt Addition (BigDecimalInt anotherDec)

Input: two BigDecimalInt objects: 'this' and 'anotherDec'

Output: a new BigDecimalInt object representing the result of the addition

1. Initialize an empty string 'result' to store the result.

2. Determine the signs of 'this' and 'anotherDec'.

3. If both 'this' and 'anotherDec' have the same sign:

a. If both are positive, set the sign of 'result' as positive ('+').

b. If both are negative, set the sign of 'result' as negative ('-').

4. Compare the lengths of 'this' and 'anotherDec', and make sure 'this' is not smaller than 'anotherDec'. If it is, swap the values.

5. Reverse the strings of 'this' and 'anotherDec'.

6. Initialize variables 's_n1' and 's_n2' to the lengths of 'this' and 'anotherDec' minus 1, respectively.

7. Initialize a 'carry' variable to 0.

8. Loop from index 0 to 's_n1':

a. Calculate the sum of the digits at index 'i' in 'this' and 'anotherDec' and add the carry.

b. Append the last digit of the sum to 'result'.

c. Update 'carry' with the carry of the sum.

9. Continue looping from 's_n1' to 's_n2':

a. Add the digit at index 'i' in 'anotherDec' to 'result' and add the carry.

b. Append the last digit of the sum to 'result'.

c. Update 'carry' with the carry of the sum.

10. If there's a carry left, append it to 'result'.

11. Reverse 'result' to correct its order.

12. Return 'result' as a new BigDecimalInt object.

Subtraction Algorithm (-):

Algorithm BigDecimalInt Subtraction (BigDecimalInt anotherDec)

Input: two BigDecimalInt objects: 'this' and 'anotherDec'

Output: a new BigDecimalInt object representing the result of the subtraction

1. Initialize an empty string 'result' to store the result.

2. Determine the signs of 'this' and 'anotherDec'.

3. If 'this' and 'anotherDec' have the same sign, perform subtraction as addition with signs adjusted.

4. If 'this' and 'anotherDec' have different signs:

a. Determine the larger number by comparing their values.

b. Remove the signs from 'this' and 'anotherDec'.

c. Reverse the strings of 'this' and 'anotherDec'.

d. Initialize variables 's_n1' and 's_n2' to the lengths of 'this' and 'anotherDec' minus 1, respectively.

e. Initialize a 'carry' variable to 0.

f. Loop from index 0 to 's_n1':

i. Calculate the difference between the digits at index 'i' in 'this' and 'anotherDec' and subtract the carry.

ii. If the result is negative, add 10 to it and set 'carry' to 1; otherwise, set 'carry' to 0.

iii. Append the result to 'result'.

g. Continue looping from 's_n1' to 's_n2':

i. Subtract the digit at index 'i' in 'anotherDec' from 'this' and subtract the carry.

ii. If the result is negative, add 10 to it and set 'carry' to 1; otherwise, set 'carry' to 0.

iii. Append the result to 'result'.

h. If 'this' was the smaller number, append a negative sign to 'result'.

i. Reverse 'result' to correct its order.

5. Return 'result' as a new BigDecimalInt object.

Less Than Comparison Algorithm (<):

Algorithm BigDecimalInt Less Than Comparison (BigDecimalInt anotherDec)

Input: two BigDecimalInt objects: 'this' and 'anotherDec'

Output: a boolean value indicating if 'this' is less than 'anotherDec'

1. Compare the signs of 'this' and 'anotherDec':

a. If 'this' is positive and 'anotherDec' is negative, return false.

b. If 'this' is negative and 'anotherDec' is positive, return true.

2. Compare the lengths of 'this' and 'anotherDec':

a. If 'this' is shorter than 'anotherDec', return true.

b. If 'this' is longer than 'anotherDec', return false.

3. Iterate through the digits of 'this' and 'anotherDec' from left to right:

a. Compare the digits at the same position.

b. If the digit in 'this' is smaller, return true.

c. If the digit in 'this' is larger, return false.

4. If the two numbers are equal, return false (they are not less than each other).

Greater Than Comparison Algorithm (>):

Algorithm BigDecimalInt Greater Than Comparison (BigDecimalInt anotherDec)

Input: two BigDecimalInt objects: 'this' and 'anotherDec'

Output: a boolean value indicating if 'this' is greater than 'anotherDec'

1. Compare the signs of 'this' and 'anotherDec':
   a. If 'this' is positive and 'anotherDec' is negative, return true.
   b. If 'this' is negative and 'anotherDec' is positive, return false.
2. Compare the lengths of 'this' and 'anotherDec':
   a. If 'this' is shorter than 'anotherDec', return false.
   b. If 'this' is longer than 'anotherDec', return true.
3. Iterate through the digits of 'this' and 'anotherDec' from left to right:
   a. Compare the digits at the same position.
   b. If the digit in 'this' is larger, return true.
   c. If the digit in 'this' is smaller, return false.
4. If the two numbers are equal, return false (they are not greater than each other).

isValidReal Algorithm:

Algorithm isValidReal (string input)

Input: a string 'input' representing a potential real number

Output: a boolean value indicating if 'input' is a valid representation of a real number

1. Initialize 'find_plus', 'find_min', 'find_space' as true to keep track of the presence of signs and spaces in the input.

2. If the first character of 'input' is a digit (0-9) and not '0', add a '+' sign to 'input'.

3. Loop through each character in 'input':

a. Check for spaces: If a