Fraction Addition and Subtraction

Given a string representing an expression of fraction addition and subtraction, you need to return the calculation result in string format. The final result should be irreducible fraction. If your final result is an integer, say `2`, you need to change it to the format of fraction that has denominator `1`. So in this case, `2` should be converted to `2/1`.

Example 1:

```
Input:"-1/2+1/2"
Output: "0/1"
```

Example 2:

```
Input:"-1/2+1/2+1/3"
Output: "1/3"
```

Example 3:

```
Input:"1/3-1/2"
Output: "-1/6"
```

Example 4:

```
Input:"5/3+1/3"
Output: "2/1"
```

Note:

1. The input string only contains `'0'` to `'9'`, `'/'`, `'+'` and `'-'`. So does the output.
2. Each fraction (input and output) has format `±numerator/denominator`. If the first input fraction or the output is positive, then `'+'` will be omitted.
3. The input only contains valid **irreducible fractions**, where the **numerator** and **denominator** of each fraction will always be in the range [1,10]. If the denominator is 1, it means this fraction is actually an integer in a fraction format defined above.
4. The number of given fractions will be in the range [1,10].
5. The numerator and denominator of the **final result** are guaranteed to be valid and in the range of 32-bit int.

Solution 1

```
public String fractionAddition(String expression) {
    String[] fracs = expression.split("(?=[-,+])"); // splits input string into ind
ividual fractions
    String res = "0/1";
    for (String frac : fracs) res = add(res, frac); // add all fractions together
    return res;
}

public String add(String frac1, String frac2) {
    int[] f1 = Stream.of(frac1.split("/")).mapToInt(Integer::parseInt).toArray(),
          f2 = Stream.of(frac2.split("/")).mapToInt(Integer::parseInt).toArray();
    int numer = f1[0]*f2[1] + f1[1]*f2[0], denom = f1[1]*f2[1];
    String sign = "";
    if (numer < 0) {sign = "-"; numer *= -1;}
    return sign + numer/gcd(numer, denom) + "/" + denom/gcd(numer, denom); // const
ruct reduced fraction
}

// Computes gcd using Euclidean algorithm
public int gcd(int x, int y) { return x == 0 || y == 0 ? x + y : gcd(y, x % y); }
```

written by compton_scatter original link here

Solution 2

```java
public class Solution {
    public String fractionAddition(String expression) {
        List<String> nums = new ArrayList<>();
        int i = 0, j = 0;
        while (j <= expression.length()) {
            if (j == expression.length() || j != i && (expression.charAt(j) == '+' |
| expression.charAt(j) == '-')) {
                if (expression.charAt(i) == '+') {
                    nums.add(expression.substring(i + 1, j));
                }
                else {
                    nums.add(expression.substring(i, j));
                }

                i = j;
            }
            j++;
        }

        String result = "0/1";

        for (String num : nums) {
            result = add(result, num);
        }

        return result;
    }

    private String add(String s1, String s2) {
        String[] sa1 = s1.split("/");
        String[] sa2 = s2.split("/");
        int n1 = Integer.parseInt(sa1[0]);
        int d1 = Integer.parseInt(sa1[1]);
        int n2 = Integer.parseInt(sa2[0]);
        int d2 = Integer.parseInt(sa2[1]);

        int n = n1 * d2 + n2 * d1;
        int d = d1 * d2;

        if (n == 0) return "0/1";

        boolean isNegative = n * d < 0;
        n = Math.abs(n);
        d = Math.abs(d);
        int gcd = getGCD(n, d);

        return (isNegative ? "-" : "") + (n / gcd) + "/" + (d / gcd);
    }

    private int getGCD(int a, int b) {
        if (a == 0 || b == 0) return a + b; // base case
        return getGCD(b, a % b);
    }
}
```

written by shawngao original link here

Solution 3

Keep the overall result in `A / B`, read the next fraction into `a / b`. Their sum is `(Ab + aB) / Bb` (but cancel their greatest common divisor).

C++:

```
string fractionAddition(string expression) {
    istringstream in(expression);
    int A = 0, B = 1, a, b;
    char _;
    while (in >> a >> _ >> b) {
        A = A * b + a * B;
        B *= b;
        int g = abs(__gcd(A, B));
        A /= g;
        B /= g;
    }
    return to_string(A) + '/' + to_string(B);
}
```

Java:

```
public String fractionAddition(String expression) {
    Scanner sc = new Scanner(expression).useDelimiter("/|(?=[-+])");
    int A = 0, B = 1;
    while (sc.hasNext()) {
        int a = sc.nextInt(), b = sc.nextInt();
        A = A * b + a * B;
        B *= b;
        int g = gcd(A, B);
        A /= g;
        B /= g;
    }
    return A + "/" + B;
}

private int gcd(int a, int b) {
    return a != 0 ? gcd(b % a, a) : Math.abs(b);
}
```

Python 3:

Added this after @lee215 reminded me about Python 3's `math.gcd` with his solution in the comments.

```
def fractionAddition(self, expression):
    ints = map(int, re.findall('[+-]?\d+', expression))
    A, B = 0, 1
    for a in ints:
        b = next(ints)
        A = A * b + a * B
        B *= b
        g = math.gcd(A, B)
        A //= g
        B //= g
    return '%d/%d' % (A, B)
```

written by StefanPochmann original link here

From Leetcoder.