프로그래머스 - 다항식 더하기

태그 1월02일

다항식 더하기
문제 설명
한 개 이상의 항의 합으로 이루어진 식을 다항식이라고 합니다. 다항식을 계산할 때는 동류항끼리 계산해 정리합니다. 덧셈으로 이루어진 다항식 polynomial이 매개변수로 주어질 때, 동류항끼리 더한 결괏값을 문자열로 return 하도록 solution 함수를 완성해보세요. 같은 식이라면 가장 짧은 수식을 return 합니다.

제한사항
0 < polynomial에 있는 수 < 100

polynomial에 변수는 'x'만 존재합니다.

polynomial은 0부터 9까지의 정수, 공백, ‘x’, ‘+'로 이루어져 있습니다.

항과 연산기호 사이에는 항상 공백이 존재합니다.

공백은 연속되지 않으며 시작이나 끝에는 공백이 없습니다.

하나의 항에서 변수가 숫자 앞에 오는 경우는 없습니다.

" + 3xx + + x7 + "와 같은 잘못된 입력은 주어지지 않습니다.

"012x + 001"처럼 0을 제외하고는 0으로 시작하는 수는 없습니다.

문자와 숫자 사이의 곱하기는 생략합니다.

polynomial에는 일차 항과 상수항만 존재합니다.

계수 1은 생략합니다.

결괏값에 상수항은 마지막에 둡니다.

0 < polynomial의 길이 < 50
입출력 예 #1

"3x + 7 + x"에서 동류항끼리 더하면 "4x + 7"입니다.
입출력 예 #2

"x + x + x"에서 동류항끼리 더하면 "3x"입니다.

```
function solution(polynomial) {
  const arrPolynomial = polynomial.split(" ");
  let arrX = arrPolynomial
    .filter((v) => v.includes("x"))
    .map((v) => (v.split("x")[0] ? v.split("x")[0] : "1"));
  let num = arrPolynomial.filter((v) => +v);
```

```
  if (arrX.length && num.length) {
    arrX = arrX.reduce((a, b) => +a + +b);
    num = num.reduce((a, b) => +a + +b);
    if (arrX === "1") arrX = "";
    return `${arrX + ""}x + ${num + ""}`;
  } else {
    if (arrX.length) {
      arrX = arrX.reduce((a, b) => +a + +b);
      if (arrX === "1") arrX = "";
      return `${arrX + ""}x`;
    }
    if (num.length) {
      num = num.reduce((a, b) => +a + +b);
      return num + "";
    }
  }
}
```

```
각 값별로 인덱스를 사용하기 위해서 배열(arrPolynomial)로 바꿔(.split(" "))줍니다.
 * polynomial[0] => 3,  arrPolynomial[0] => 3x
 *
 * 주어진 문자열은 "x"와 "num" 그리고 "+" 입니다
 * 단 사칙연산은 "+"만 존재 하므로 제외하고
 * x와 num를 분리합니다.
 *
 * x
 * arrPolynomial에서 "x"를 포함하는 값을 뽑아낸 배열을 만듭니다.
 * .filter((v) => v.includes("x"))
 * 위 배열에서 각 값을 "x"를 기준으로 나눈 배열로 만들고
 * 나눈 배열의 0번 인덱스를 리턴하는데
 * 0번 인덱스가 ""이 아니라면 v.split("x")[0] ""이라면 "1"을 리턴합니다.
 * .map((v) => (v.split("x")[0] ? v.split("x")[0] : "1"));
 *
 * num
 * arrPolynomial에서 숫자로 바꾸었을때 참인 값을 배열로 만듭니다.
 * arrPolynomial.filter((v) => +v)
 *
 * 만약 arrX.length와 num.length가 둘다 존재한다면
 * 두 배열 모두 배열의 값을 숫자로 바꾸고 모두 더해주고
 * arrX = arrX.reduce((a, b) => +a + +b);
 * num = num.reduce((a, b) => +a + +b);
 * `${arrX + ""}x + ${num + ""}`를 리턴합니다.
 * 만약 두배열 중 하나만 존재할 경우
 * 존재하는 배열의 값을 모두 숫자로 바꾸고 모두 더해주고
 * 형식에 맞게 리턴합니다.
 * 추가 arrX가 "1"인 경우 1x로 표기가 되어서
 * arrX가 1인 경우는 arrX를 ""로 바꿔 줍니다.
```

```
function solution(polynomial) {
    const arr = polynomial.split(" + ");
    const xNum = arr
                .filter(n => n.includes("x"))
```

```
                .map(n => n.replace('x', '') || '1')
                .reduce((acc, cur) => acc + parseInt(cur, 10), 0);
    const num = arr
                .filter(n => !isNaN(n))
                .reduce((acc, cur) => acc + parseInt(cur, 10), 0);

    let answer = [];
    if(xNum) answer.push(`${xNum === 1 ? "" : xNum}x`);
    if(num) answer.push(num);

    return answer.join(" + ");
}
```

먼저 다항식을 항의 배열로 분할하고 배열을 필터링하여 x를 포함하는 항과 상수 항을 분리합니다. 그런 다음 x 항을 매핑하여 계수를 추출하고 두 배열을 축소하여 x 항과 상수 항의 합을 얻습니다.

마지막으로 함수는 x 항(0이 아닌 경우)의 합을 상수 항(0이 아니고 x 항의 합이 0인 경우)과 연결하여 결과 다항식 문자열을 만듭니다. x 항과 상수 항의 합이 모두 0이면 함수는 빈 문자열을 반환합니다.

```
class Solution {
    public String solution(String polynomial) {
        int xCount = 0;
        int num = 0;

        for (String s : polynomial.split(" ")) {
            if (s.contains("x")) {
                xCount += s.equals("x") ? 1 : Integer.parseInt(s.replaceAll("x", ""));
            } else if (!s.equals("+")) {
                num += Integer.parseInt(s);
            }
        }
        return (xCount != 0 ? xCount > 1 ? xCount + "x" : "x" : "") + (num != 0 ? (xCo
unt != 0 ? " + " : "") + num : xCount == 0 ? "0" : "");
    }
}
```

이 방법은 먼저 두 개의 변수 xCount와 num을 초기화하여 각각 총 x 항 수와 상수 항을 저장합니다. 그런 다음 다항식을 용어 배열로 분할하고 각 용어를 반복합니다.

용어에 x가 포함되어 있으면 계수가 추출되어 xCount에 추가됩니다. 항이 상수이면 num에 추가됩니다.

마지막으로 메서드는 xCount 값(0이 아닌 경우)을 num 값(0이 아니고 xCount가 0인 경우)과 연결하여 결과 다항식 문자열을 만듭니다. xCount와 num이 모두 0이면 메서드는 "0"을 반환합니다.

```
import java.util.StringTokenizer;

class Solution {
```

```
    public String solution(String polynomial) {
        StringTokenizer st = new StringTokenizer(polynomial, " + ");
        StringBuilder sb = new StringBuilder();

        int xsum = 0;
        int sum = 0;
        while (st.hasMoreTokens()) {
            String str = st.nextToken();

            if (str.contains("x")) {
                String x = str.replace("x", "");
                System.out.println(x);
                if (x.isBlank()) {
                    xsum += 1;
                } else {
                    xsum += Integer.parseInt(x);
                }
            } else {
                sum += Integer.parseInt(str);
            }
        }

        if (xsum == 0) {
            sb.append(sum);
        } else if (sum == 0) {
            if (xsum == 1) {
                sb.append("x");
            } else {
                sb.append(xsum).append("x");
            }
        } else {
            if (xsum == 1) {
                sb.append("x").append(" + ").append(sum);
            } else {
                sb.append(xsum).append("x").append(" + ").append(sum);
            }

        }

        return sb.toString();
    }
}
```

StringTokenizer 개체를 사용하여 다항식을 개별 용어로 분할하고 StringBuilder 개체를 사용하여 결과 다항식 문자열을 만듭니다. 두 개의 변수 xsum과 sum을 초기화하여 x 항과 상수 항의 총 합을 각각 저장합니다.

그런 다음 메서드는 항을 반복하고 계수와 상수 항을 추출합니다. 용어에 x가 포함되어 있으면 계수가 추출되어 xsum에 추가됩니다. 항이 상수이면 합계에 추가됩니다.

마지막으로 메서드는 xsum 값(0이 아닌 경우)을 합계 값(0이 아니고 xsum이 0인 경우)과 연결하여 결과 다항식 문자열을 만듭니다. xsum과 sum이 모두 0이면 메서드는 "0"을 반환합니다.