Job No: 08

Job Name: **Write a program to eliminate left recursion from a production of a grammar.**

Theory:

A production of grammar is said to have left recursion if the leftmost variable of its RHS is same as variable of LHS.

If we have the left-recursive pair of productions:
$A \rightarrow A\alpha / \beta$
where β does not begin with an A.

Then, we can eliminate left recursion by replacing the pair of productions with:
$A \rightarrow \beta A'$
$A' \rightarrow \alpha A' / \in$

Code:

```
const RemoveLeftRecursion = (nonTerminal,productionRule) => {
    let output1='';
    let output2='';
    let productions = productionRule.split('|').filter(e => e!=='#');

    for(let production of productions) {
        if(!production.startsWith(nonTerminal)) {
            output1 += `${production}${nonTerminal}'|`;
            console.log(`Production ${productions.indexOf(production)+1}
                        does not have left recursion`);
        } else {
            output2 += production.replace(nonTerminal,'')
                                 .concat(`${nonTerminal}'|`);
            console.log(`Production ${productions.indexOf(production)+1}
                        has left recursion`);
        }
    };

    console.log(`The output is:`);
    output1 = output1.slice(0, output1.length - 1);
    console.log(`${nonTerminal} --> ${output1}`);
```

```
    console.log(`${nonTerminal}' --> ${output2}#`);
}

let nonTerminal = 'E';
let productions = 'Ea|Eb|z';
console.log(`The given grammer is: ${nonTerminal} --> ${productions}`);

RemoveLeftRecursion(nonTerminal,productions);
```

Input/Output:

```
The given grammer is: E --> Ea|Eb|z
Production 1 has left recursion
Production 2 has left recursion
Production 3 does not have left recursion
The output is:
E --> zE'
E' --> aE'|bE'|#
```

Job No: 08

Job Name: **Write a program to eliminate left recursion from a production of a grammar.**

Theory:

A production of grammar is said to have left recursion if the leftmost variable of its RHS is same as variable of LHS.

If we have the left-recursive pair of productions:
A → Aα / β
where β does not begin with an A.

Then, we can eliminate left recursion by replacing the pair of productions with:
A → βA′
A′ → αA′ / ∈

Code:

```
const deleteLeftRecursion = (parentNonTerminal,productionRule) => {
    let result1='';
    let result2='';
    let productionList = productionRule.split('|').filter(e => e!=='#');

    for(let production of productionList) {
        if(!production.startsWith(parentNonTerminal)){
            result1 += `${production}${parentNonTerminal}'|`;
            console.log(`Production
                        ${productionList.indexOf(production)+1}
                        does not have left recursion`);
        }else{
           result2 += production.replace(parentNonTerminal,'')
                                .concat(`${parentNonTerminal}'|`);
           console.log(`Production
                       ${productionList.indexOf(production)+1}
                       has left recursion`);
        }
    };
    console.log(`The output is:`);
    result1 = result1.slice(0, result1.length - 1);
```

```
    console.log(`${parentNonTerminal} --> ${result1}`);
    console.log(`${parentNonTerminal}' --> ${result2}#`);
}

let parentNonTerminal = 'F';
let productionList = 'FBd|Fa|a';
console.log(`The given grammer is:
            ${parentNonTerminal} --> ${productionList}`);

deleteLeftRecursion(parentNonTerminal,productionList);
```

Input/Output:

```
The given grammer is: F --> FBd|Fa|a
Production 1 has left recursion
Production 2 has left recursion
Production 3 does not have left recursion
The output is:
F --> aF'
F' --> BdF'|aF'|#
```