Ishita Hardasmalani

C14-2103058

EXPERIMENT NO.10

Aim : Write a program to eliminate left recursion from the given grammar

Code:

```
def removeLeftRecursion(rulesDiction):
    store = {}
    for lhs in rulesDiction:
        alphaRules = []
        betaRules = []
        allrhs = rulesDiction[lhs]
        for subrhs in allrhs:
            if subrhs[0] == lhs:
                alphaRules.append(subrhs[1:])
            else:
                betaRules.append(subrhs)
        if len(alphaRules) != 0:
            lhs_ = lhs + "'"
            while lhs_ in rulesDiction.keys() or lhs_ in store.keys():
                lhs_ += "'"
            for b in range(0, len(betaRules)):
                betaRules[b].append(lhs_)
            rulesDiction[lhs] = betaRules
            for a in range(0, len(alphaRules)):
                alphaRules[a].append(lhs_)
            alphaRules.append(['#'])
            store[lhs_] = alphaRules
```

```
    for left in store:
        rulesDiction[left] = store[left]
    return rulesDiction

# Example grammar rules
rulesDiction = {
    'A': [['A', 'a'], ['A', 'b'], ['c']],
    'B': [['B', 'x'], ['y']]
}
# Apply left recursion elimination
result = removeLeftRecursion(rulesDiction)
# Print the modified rules
for key, value in result.items():
    print(f"{key} -> {value}")
```

Output: