```python
from copy import deepcopy
from re import sub

O_REL = ['A', 'B', 'C', 'D']
FD = [
    ['AB', 'B'],
    ['B', 'C'],
    ['C', 'D'],
]
NEW_REL = [
    ['A', 'B'],
    ['B', 'C'],
    ['C', 'D'],
]

def in_count(_itr: list[str], elem: str):
    return sum([1 for i in _itr if elem in i])

def initialize(old_rel, new_rel):
    table: list[list[str]] = [["" for _ in old_rel] for _ in new_rel]
    for row_index, row in enumerate(table):
        for col_index, col in enumerate(row):
            if old_rel[col_index] in new_rel[row_index]:
                table[row_index][col_index] = f"a({col_index})"
            else:
                table[row_index][col_index] = f"b({row_index},{col_index})"
    return table

def attrs_intersection(attrs: list[str]) -> str:
    for s in attrs:
    # remove content inside parenthesis
        s = sub(r'\([^)]*\)', '', s)
        s = s.replace(" ", "")
    return 'a' if all([True if 'a' in attr else False for attr in attrs]) else
'b'

def get_lhs_values(dep, table, old_rel):
    check_lhs: list[str] = []
    compound_attr: list[list[str]] = []
    for attr in dep[0]:
        compound_attr.append([row[old_rel.index(attr)] for row in table])
    for i, c_attr in enumerate(map(list, zip(*compound_attr))):
        check_lhs.append(attrs_intersection(c_attr))
        # at this point we have a comparison list, that we can use to check for
the step 2 conditions
    return check_lhs

def print_table(table: list[list[str]]):
```

```
    pretty_table = ""
    for row in table:
        pretty_table += "|"
        for columns in row:
            pretty_table += f"{columns}\t|"
        pretty_table += "\n"

    length = len(pretty_table.split("\n")[0].expandtabs())
    print("_" * length)
    print(pretty_table)

def woolman_algorithm_test(old_rel: list[str], new_rel: list[list[str]], fd:
list[list[str]]) -> bool:
    # Step 1: Initialize
    table = initialize(old_rel, new_rel)
    print("Initial Table: ")
    print_table(table)
    while True:
        compare_table = deepcopy(table)
        for dep in fd:
            print(f"Checking FD: {dep[0]} -> {dep[1]}")
            check_lhs = get_lhs_values(dep, table, old_rel)

            rows_to_change = [i for i, x in enumerate(check_lhs) if x == "a"]
            columns_to_change = [old_rel.index(attr) for attr in dep[1]]
            column_values = ["X" for _ in columns_to_change]

            if check_lhs.count("a") <= 1:
                print("No common attributes found")
                continue

            for column_index in columns_to_change:
                test_column = [row[column_index] for row in table if
table.index(row) in rows_to_change]
                # Find if there is "A" in the columns we wish to update, else get
the first "B" value
                if in_count(test_column, "a") >= 1:
                    column_values[columns_to_change.index(column_index)] =
f"a({column_index})"
                else:
                    column_values[columns_to_change.index(column_index)] =
test_column[0]

            # Do da update
            for row_index, row in enumerate(table):
                for col_index, col in enumerate(row):
                    if col_index in columns_to_change and row_index in
rows_to_change:
```

```
                        table[row_index][col_index] =
column_values[columns_to_change.index(col_index)]
            # Print Table
            print_table(table)

        if compare_table == table:
            for row in table:
                if in_count(row, "a") == len(row):
                    print(f"Found a row with all 'a' values. Row:
{table.index(row) + 1}")
                    return True
            print("No more changes possible.")
            return False


if __name__ == '__main__':
    if woolman_algorithm_test(O_REL, NEW_REL, FD):
        print("The given relation decomposition is LOSSLESS.")
    else:
        print("The given relation decomposition is LOSSY.")
```

OUTPUT

```
Initial Table:

---------------------------------
|a(0)   |a(1)   |b(0,2) |b(0,3) |
|b(1,0) |a(1)   |a(2)   |b(1,3) |
|b(2,0) |b(2,1) |a(2)   |a(3)   |


Checking FD: AB -> B
No common attributes found
Checking FD: B -> C

---------------------------------
|a(0)   |a(1)   |a(2)   |b(0,3) |
|b(1,0) |a(1)   |a(2)   |b(1,3) |
|b(2,0) |b(2,1) |a(2)   |a(3)   |


Checking FD: C -> D

---------------------------------
|a(0)   |a(1)   |a(2)   |a(3)   |
|b(1,0) |a(1)   |a(2)   |a(3)   |
```

```
|b(2,0) |b(2,1) |a(2)   |a(3)   |

Checking FD: AB -> B
No common attributes found
Checking FD: B -> C
---------------------------------
|a(0)   |a(1)   |a(2)   |a(3)   |
|b(1,0) |a(1)   |a(2)   |a(3)   |
|b(2,0) |b(2,1) |a(2)   |a(3)   |

Checking FD: C -> D
---------------------------------
|a(0)   |a(1)   |a(2)   |a(3)   |
|b(1,0) |a(1)   |a(2)   |a(3)   |
|b(2,0) |b(2,1) |a(2)   |a(3)   |

Found a row with all 'a' values. Row: 1
The given relation decomposition is LOSSLESS.
```