LAB 7: Implement unification in first order logic:

```
class Term:
    def __init__(self, name, args=None):
        self.name = name
        self.args = args if args else []

    def __repr__(self):
        if self.args:
            return f"{self.name}({', '.join(map(str, self.args))})"
        return self.name

    def is_variable(self):
        return not self.args


class Variable(Term):
    def __init__(self, name):
        super().__init__(name)

    def __repr__(self):
        return f"{self.name}"


def occurs_check(var, term):
    """Checks if a variable occurs in a term"""
    if isinstance(term, Variable) and term.name == var.name:
        return True
    if isinstance(term, Term):
        return any(occurs_check(var, arg) for arg in term.args)
    return False


def unify(term1, term2, substitution=None):
    """Unifies two terms and returns the substitution"""
```

```
    if substitution is None:
        substitution = {}

    # Base cases:
    if isinstance(term1, Variable):
        if term1.name == term2.name:
            return substitution  # No change needed
        if occurs_check(term1, term2):
            raise Exception(f"Occurs check fails: {term1} in {term2}")
        substitution[term1.name] = term2
        return substitution

    if isinstance(term2, Variable):
        if term1.name == term2.name:
            return substitution  # No change needed
        if occurs_check(term2, term1):
            raise Exception(f"Occurs check fails: {term2} in {term1}")
        substitution[term2.name] = term1
        return substitution

    if term1.name == term2.name:
        # Both are compound terms of the same function, unify their arguments
        if len(term1.args) != len(term2.args):
            raise Exception(f"Arity mismatch: {term1} and {term2}")
        for arg1, arg2 in zip(term1.args, term2.args):
            substitution = unify(arg1, arg2, substitution)
        return substitution

    raise Exception(f"Cannot unify {term1} with {term2}")


# Test examples
try:
    # Unifying terms: f(x, y) with f(a, b)
```

```
    x = Variable('x')
    y = Variable('y')
    a = Term('a')
    b = Term('b')
    term1 = Term('f', [x, y])
    term2 = Term('f', [a, b])

    substitution = unify(term1, term2)
    print("Unification successful! Substitution:", substitution)

    # Unifying terms: f(x) with g(x)
    term1 = Term('f', [x])
    term2 = Term('g', [x])

    substitution = unify(term1, term2)
    print("Unification successful! Substitution:", substitution)

except Exception as e:
    print(f"Unification failed: {e}")
```

output:

```
    # Unifying terms: f(x) with g(x)
    term1 = Term('f', [x])
    term2 = Term('g', [x])

    substitution = unify(term1, term2)
    print("Unification successful! Substitution:", substitution)

except Exception as e:
    print(f"Unification failed: {e}")
```

```
Unification successful! Substitution: {'x': a, 'y': b}
Unification failed: Cannot unify f(x) with g(x)
```