Implement unification in first order logic

```
class UnificationError(Exception):
    pass

def occurs_check(var, term, subst):
    """Helper function to check if a variable occurs in a term."""
    if var == term:
        return True
    elif isinstance(term, (list, tuple)):
        return any(occurs_check(var, t, subst) for t in term)
    elif isinstance(term, str) and term in subst:
        return occurs_check(var, subst[term], subst)
    return False

def is_variable(term):
    """Check if a term is a variable (starts with ?)."""
    return isinstance(term, str) and term.startswith('?')

def unify(psi1, psi2, subst=None):
    """
    Unify two terms psi1 and psi2 with an optional initial
substitution.
    """
    if subst is None:
        subst = {}

    # Step 1: Handle the base cases
    if psi1 == psi2:  # Identical terms, no substitution needed
        return subst

    # Case where psi1 or psi2 is a variable
    elif is_variable(psi1):  # psi1 is a variable
        if psi1 in subst:  # Apply substitution to psi1
            return unify(subst[psi1], psi2, subst)
        elif occurs_check(psi1, psi2, subst):  # Occurs check
            raise UnificationError(f"Occurs check failed: {psi1} in
{psi2}")
        else:
            subst[psi1] = psi2  # Perform the substitution
            return subst

    elif is_variable(psi2):  # psi2 is a variable
        if psi2 in subst:  # Apply substitution to psi2
            return unify(psi1, subst[psi2], subst)
        elif occurs_check(psi2, psi1, subst):  # Occurs check
            raise UnificationError(f"Occurs check failed: {psi2} in
{psi1}")
```

```
        else:
            subst[psi2] = psi1  # Perform the substitution
            return subst

    # Step 2: Check for predicate symbols (initial symbol) mismatch
    elif isinstance(psi1, list) and isinstance(psi2, list):
        if psi1[0] != psi2[0]:  # Compare the predicate symbols
            raise UnificationError(f"Predicate symbols don't match:
{psi1[0]} != {psi2[0]}")

        # Step 3: Check if they have the same number of arguments
        if len(psi1) != len(psi2):
            raise UnificationError(f"Argument lengths don't match:
{len(psi1)} != {len(psi2)}")

        # Step 5: Iterate through the arguments (subterms) to unify
each
        for arg1, arg2 in zip(psi1[1:], psi2[1:]):  # Skip the
predicate symbol (first element)
            subst = unify(arg1, arg2, subst)

        # Step 6: Return the final substitution
        return subst

    # If none of the above cases match, return failure
    else:
        raise UnificationError(f"Cannot unify {psi1} with {psi2}")

# Example usage:
def get_input():
    try:
        term1 = eval(input("Enter the first term (e.g., ['P', 'b', 'x',
['f', ['g', 'z']]]): "))
        term2 = eval(input("Enter the second term (e.g., ['P', 'z',
['f', 'y'], ['f', 'y']]): "))
        substitution = unify(term1, term2)
        print("Unification successful!")
        print("Substitution:", substitution)
    except UnificationError as e:
        print("Unification failed:", e)
    except Exception as e:
        print("Invalid input or error:", e)

# Test the unification process
get_input()
```

Output

```
Enter the first term (e.g., ['P', 'b', 'x', ['f', ['g', 'z']]]):
['P','b','?x',['f',['g','?z']]]
Enter the second term (e.g., ['P', 'z', ['f', 'y'], ['f', 'y']]):
['P','?z',['f','?y'],['f','?y']]
Unification successful!
    Substitution: {'?z': 'b', '?x': ['f', '?y'], '?y': ['g', '?z']}
```