Implementation of Unification in First Order Logic (FOL)

```
class UnificationError(Exception):
pass
def occurs_check(var, term, subst):
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):
return isinstance(term, str) and term.startswith('?')
def unify(psi1, psi2, subst=None):
if subst is None:
subst = {}
if psi1 == psi2:
return subst
elif is_variable(psi1):
if psi1 in subst:
return unify(subst[psi1], psi2, subst)
elif occurs_check(psi1, psi2, subst):
raise UnificationError(f"Occurs check failed: {psi1} in {psi2}")
else:
subst[psi1] = psi2
```

```
return subst
elif is_variable(psi2):
if psi2 in subst:
return unify(psi1, subst[psi2], subst)
elif occurs_check(psi2, psi1, subst):
raise UnificationError(f"Occurs check failed: {psi2} in {psi1}")
else:
subst[psi2] = psi1
return subst
elif isinstance(psi1, list) and isinstance(psi2, list):
if psi1[0] != psi2[0]:
raise UnificationError(f"Predicate symbols don't match: {psi1[0]} != {psi2[0]}")
if len(psi1) != len(psi2):
raise UnificationError(f"Argument lengths don't match: {len(psi1)} !=
{len(psi2)}")
for arg1, arg2 in zip(psi1[1:], psi2[1:]):  # Skip the predicate symbol (first
element)
subst = unify(arg1, arg2, subst)
return subst
else:
raise UnificationError(f"Cannot unify {psi1} with {psi2}")
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)
get_input()
class UnificationError(Exception):
pass
def occurs_check(var, term, subst):
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):
return isinstance(term, str) and term.startswith('?')
def unify(psi1, psi2, subst=None):
if subst is None:
subst = {}
if psi1 == psi2:
return subst
```

```
elif is_variable(psi1):
if psi1 in subst:
return unify(subst[psi1], psi2, subst)
elif occurs_check(psi1, psi2, subst):
raise UnificationError(f"Occurs check failed: {psi1} in {psi2}")
else:
subst[psi1] = psi2
return subst
elif is_variable(psi2):
if psi2 in subst:
return unify(psi1, subst[psi2], subst)
elif occurs_check(psi2, psi1, subst):
raise UnificationError(f"Occurs check failed: {psi2} in {psi1}")
else:
subst[psi2] = psi1
return subst
elif isinstance(psi1, list) and isinstance(psi2, list):
if psi1[0] != psi2[0]:
raise UnificationError(f"Predicate symbols don't match: {psi1[0]} != {psi2[0]}")
if len(psi1) != len(psi2):
raise UnificationError(f"Argument lengths don't match: {len(psi1)} !=
{len(psi2)}")
for arg1, arg2 in zip(psi1[1:], psi2[1:]):  # Skip the predicate symbol (first
element)
subst = unify(arg1, arg2, subst)
```

```
return subst
else:
raise UnificationError(f"Cannot unify {psi1} with {psi2}")
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)
get_input()
```

Output:

1.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']}

2. Enter the first term (e.g., ['P', 'b', 'x', ['f', ['g', 'z']]]): ['P',['f',['a']],['g',['?y']]]

Enter the second term (e.g., ['P', 'z', ['f', 'y'], ['f', 'y']]): ['P','?x','?x']

Unification failed: Predicate symbols don't match: g != f