```
"""
PRACTICAL LAB: Sentence → Predicate → Clause Form Converter
============================================================
"""

# ──────────────────────────────────────────────
# STEP 1: Input sentence
# ──────────────────────────────────────────────
print("\nSTEP 1: Enter Your Sentence")
sentence = input("Enter sentence: ")


# ──────────────────────────────────────────────
# STEP 2: Define predicates
# ──────────────────────────────────────────────
print("\n\nSTEP 2: Define Predicates")
print("Enter predicates (press Enter when done)\n")

predicates = [] ¬
i = 1

while True:
    phrase = input(f"Predicate {i} phrase (or Enter to finish): ")

    # If the student pressed Enter without typing, stop the loop
    if phrase == "":
        break

    name = input(f"Predicate {i} name: ")

    # Store phrase and name together as a tuple, then add to list
    single_predicate = (phrase, name)
    predicates.append(single_predicate)

    i = i + 1

# ──────────────────────────────────────────────
# STEP 3: Build Predicate Form
# ──────────────────────────────────────────────
print("\n\nSTEP 3: Predicate Form")

# Decide how many predicates are CONDITIONS vs RESULTS
if len(predicates) >= 2:
    num_conditions = int(input(f"How many are conditions (1-{len(predicates)
- 1})? "))
else:
    num_conditions = len(predicates)

# Build the CONDITIONS list (left side of the arrow)
conditions = []
for index in range(num_conditions):
    predicate_name = predicates[index][1]   # [1] gets the name part of the
tuple
    conditions.append(predicate_name)

# Build the RESULTS list (right side of the arrow)
results = []
```

```
for index in range(num_conditions, len(predicates)):
    predicate_name = predicates[index][1]
    results.append(predicate_name)

# ── Build the condition string ──
if len(conditions) > 1:
    # Join multiple conditions with AND symbol
    condition_parts = []
    for c in conditions:
        condition_parts.append(f"{c}(x)")
    cond_str = " ∧ ".join(condition_parts)
else:
    cond_str = f"{conditions[0]}(x)"

# ── Build the result string ──
result_parts = []
for r in results:
    result_parts.append(f"{r}(x)")
res_str = " ∧ ".join(result_parts)

# ── Assemble the full predicate formula ──
if results:
    # There IS an implication (conditions → results)
    predicate_form = f"∀x ({cond_str} → {res_str})"
else:
    # No implication, just the conditions alone
    predicate_form = f"∀x ({cond_str})"

print(f"\n Predicate Form:")
print(f"  {predicate_form}")


# ──────────────────────────────────────────────
# STEP 4: Convert to Clause Form
# ──────────────────────────────────────────────
print("\n\nSTEP 4: Convert to Clause Form")

if "→" in predicate_form:
    print("\nConversion steps:")

    # ── Step 4.1: Eliminate implication ──
    print("\n1. Eliminate → (A → B becomes ¬A ∨ B)")

    if len(conditions) > 1:
        # Wrap the whole condition group in ¬(...)
        inner_parts = []
        for c in conditions:
            inner_parts.append(f"{c}(x)")
        inner_str = " ∧ ".join(inner_parts)
        neg_cond = "¬(" + inner_str + ")"
    else:
        neg_cond = f"¬{conditions[0]}(x)"

    step1 = f"∀x ({neg_cond} ∨ {res_str})"
    print(f"   {step1}")
```

```
    # ── Step 4.2: Apply De Morgan's Law ──
    if len(conditions) > 1:
        print("\n2. Apply De Morgan's Law: ¬(A ∧ B) becomes (¬A ∨ ¬B)")

        neg_parts = []
        for c in conditions:
            neg_parts.append(f"¬{c}(x)")
        neg_str = " ∨ ".join(neg_parts)

        step2 = f"∀x ({neg_str} ∨ {res_str})"
        print(f"   {step2}")
    else:
        step2 = step1

    # ── Step 4.3: Drop universal quantifier ──
    print("\n3. Drop ∀x (implicit in clauses)")

    clause_parts = []

    # Add negated conditions
    for c in conditions:
        clause_parts.append(f"¬{c}(x)")

    # Add results (no negation)
    for r in results:
        clause_parts.append(f"{r}(x)")

    clause_form = " ∨ ".join(clause_parts)
    print(f"   {clause_form}")

else:
    # No implication — already in clause form
    clause_parts = []
    for c in conditions:
        clause_parts.append(f"{c}(x)")

    clause_form = " ∧ ".join(clause_parts)
    print(f"\n✓ Already in clause form: {clause_form}")


# ──────────────────────────────────────────────
# FINAL RESULTS
# ──────────────────────────────────────────────
print("\nFINAL RESULTS")

print(f"\n1. English Sentence:")
print(f"   {sentence}")

print(f"\n2. Predicate Form:")
print(f"   {predicate_form}")

print(f"\n3. Clause Form:")
print(f"   {clause_form}")

print("\nConversion is complete!")
```