Aim:

To design an ER model for an Irrigation System and perform EER to Object Database (ODB) mapping using object-oriented concepts.

⚙ Algorithm (Short Version):

1. Identify entities and their attributes (Farmer, Field, Water_Source, Crop, etc.).
2. Define relationships between entities (Owns, Cultivates, Uses, etc.).
3. Draw the ER diagram.
4. Extend ER into EER using specialization and aggregation.
5. Map entities to classes, and relationships to object references/lists.
6. Use inheritance for specialized entities (e.g., Borewell, Canal, Tank).
7. Implement and display sample output.

Program:

```
# --- EER to ODB Mapping for Irrigation System ---


class Farmer:
  def _init_(self, id, name):
    self.id = id
    self.name = name
    self.fields = []


class Field:
  def _init_(self, id, location):
    self.id = id
    self.location = location
    self.water_source = None
    self.crops = []


class Water_Source:
```

```
    def _init_(self, id, type):
        self.id = id
        self.type = type

class Crop:
    def _init_(self, id, name):
        self.id = id
        self.name = name

# --- Demonstration ---
farmer1 = Farmer(1, "Ramesh")
field1 = Field(101, "Pollachi")
source1 = Water_Source(201, "Borewell")
crop1 = Crop(301, "Rice")

# Mapping
farmer1.fields.append(field1)
field1.water_source = source1
field1.crops.append(crop1)

# --- Output ---
print("Farmer:", farmer1.name)
print("Field:", field1.location)
print("Water Source:", field1.water_source.type)
print("Crop:", field1.crops[0].name)
```

**Output:**

Farmer: Ramesh

Field: Pollachi

Water Source: Borewell

Crop: Rice

Result:

The ER model for the Irrigation System was designed and successfully mapped to an Object Database using classes and relationships.