Q1: Implement the Stack ADT in **Python** using classes. Your class should have following methods:

Method	Description
`push(x)`	Adds `x` to the top of the stack
`pop()`	Removes and returns the top element
`peek()`	Returns top element (without removing it)
`is_empty()`	Returns `True` if stack is empty
`size()`	Returns number of elements in stack

Create a main program to test the following:

1. Push 3 elements (e.g., 10, 20, 30)
2. Print current stack
3. Peek at top element
4. Pop one element
5. Print stack and size again
6. Check if stack is empty
7. Attempt to `pop()` from an empty stack and catch the error

Q2 : Implement the Queue ADT in Python using classes. Your class should have the following methods:

Method	Description
`enqueue(x)`	Adds `x` to the rear of the queue
`dequeue()`	Removes and returns the front element
`peek()`	Returns the front element (without removing it)
`is_empty()`	Returns `True` if the queue is empty
`size()`	Returns the number of elements in the queue

Create a main program to test the following:

1. Enqueue 3 elements (e.g., 10, 20, 30)
2. Print current queue
3. Peek at the front element
4. Dequeue one element
5. Print queue and size again
6. Check if queue is empty
7. Attempt to `dequeue()` from an empty queue and catch the error

Q3: Implement the Set ADT in Python using classes. Your class should have the following methods:

Method	Description
`add(x)`	Adds `x` to the set (if not already present)
`remove(x)`	Removes `x` from the set
`contains(x)`	Returns `True` if `x` exists in the set
`is_empty()`	Returns `True` if the set has no elements
`size()`	Returns the number of elements in the set
`union(other_set)`	Returns a new set that is the union of this set and another set
`intersection(other_set)`	Returns a new set with elements common to both sets
`difference(other_set)`	Returns a new set with elements in this set but not in the other

Create a main program to test the following:

1. Add 3 elements (e.g., 10, 20, 30) to the set
2. Try adding duplicates (e.g., 20, 10)
3. Print the current set and its size
4. Check if specific elements (e.g., 20 and 40) are present
5. Remove an element (e.g., 20)
6. Attempt to remove an element that doesn't exist (e.g., 100), and catch the error
7. Check if the set is empty
8. Create another set with elements (e.g., 30, 40, 50) and test:
    a. `union()`
    b. `intersection()`
    c. `difference()`