Q1.Create a generic class Box<T> that:

- Stores a single value of type T
- Has:
  - void set(T value)
  - T get()
- Override toString() to print the value

Example:

```
Box<String> box1 = new Box<>();
box1.set("Hello");

Box<Integer> box2 = new Box<>();
box2.set(100);

System.out.println(box1.get());  // Hello
System.out.println(box2.get());  // 100
```

Q2.Write a generic static method:

```
public static <T extends Comparable<T>> T findMax(T a, T b, T c)
```

Requirements:

- Return the largest of three values
- Must work with:
  - Integer
  - Double
  - String
  - Any custom class implementing Comparable

Example:

```
findMax(10, 20, 5);         // 20
findMax("A", "Z", "M");      // Z
```

Q3. Implement a generic class:

class Stack<T>

Requirements:

- Use an internal array
- Methods:
  - void push(T value)
  - T pop()
  - T peek()
  - boolean isEmpty()
- Resize array when full (dynamic resizing)

Example :

```
Stack<String> stack = new Stack<>();
stack.push("A");
stack.push("B");

System.out.println(stack.pop());  // B
```

Q4. Implement a generic method:

```
public static <T> void copy(
   List<? super T> destination,
   List<? extends T> source)
```

Requirements:

- Copy all elements from source to destination
- Follow PECS rule
- Should work for:

```
List<Integer> integers = List.of(1, 2, 3);
List<Number> numbers = new ArrayList<>();

copy(numbers, integers);
```

After copy, numbers should contain [1, 2, 3].