Java class/method name being tested: Date.isValid			
Test Case #	Requirement	Test Description and Input Data	Expected Output/Result
Test Case #1	The method must prevent invalid dates such as 31 days in a month with 30 or less days.	Create an instance of Date with valid month and year but invalid day. Test data: “test data: “3/39/3022”	False/False
Test Case #2	Number of days in February for a non- leap year shall be 28.	Create an instance of Date with the month = 2, day > 28, and the year is a non-leap year Test Data: “02/29/2022”	False/False
Test Case #3	The method must prevent invalid months from being entered in keeping the months STRICTLY between 1-12.	Create an instance of Date with valid date and year but invalid month. Test Data: “35/05/2022”	False/False
Test Case #4	The method must prevent a month from being entered in as all zeros.	Create an instance of Date with invalid month but valid date and year but invalid day. Test Dates: “00/09/2000”	False/False
Test Case #5	The method must prevent a date from being entered in as all zeros.	Create an instance of Date with Invalid day but valid month and year but invalid day. Test: “06/00/2000”	False/False
Test Case #6	The method must prevent a year from being entered in as all zeros.	Create an instance of Date with Invalid day but valid month and year but invalid day. Test: “06/15/0000”	False/False
Test Case #7	The method must prevent day, month and year from being entered in as all zeros.	Create an instance of Date with All zeros being imputed in days, month, year. Test: “00/00/0000”	False/False
Test Case #8	The method must return true for February 29th if the year is leap year.	Create an instance of Date with Feb 29th, leap year. Test: "2/29/2024"	True/True

Test Case #9	Check if it returns true for normal date.	Create an instance with normal date. Test: "3/1/1997"	True/True

Java class/method name being tested: Member.compareTo			
Test Case #	Requirement	Test Description and Input Data	Expected Output/Result
Test Case #1	The method should return 0 if it compares same name.	Create two Member instances with same name. Test: Member 1 = "John Doe" Member 2 = "John Doe"	0/0
Test Case #2	The method needs to check first name's order if both have same last name	Create two Member instances with same last name and different first name. Test: Member 1 = "John Doe" Member 2 = "Jane Doe"	1/1
Test Case #3	The method needs to compare the last name first.	Create two Member instances. Member 1 has earlier first name but later last name. Test: Member 1 = "April March" Member 2 = "Mary Lindsey"	1/1
Test Case #4	The method should order the Member instances with same first name and different last names.	Create two Member instances with same first name and different last name. Test: Member 1 = "John Doe" Member 2 = "John Doa"	1/1
Test Case #5	Check again if the method compares the	Create two Member instances. Member 1 has later	-1/-1

	last name first and then first name.	first name, but earlier last name. Test: Member 1 = "Roy Brooks" Member 2 = "Bill Scalan"	