class name: Date
method signature: public Boolean isValid() {} // check if a given date is a valid calendar date

Test Case #	Requirement	Test description and Input Data	Expected result/output
1	The method shall return false for February 29 on a non-leap year	Create an instance of Date with February 29 on a non-leap year **Test input:** "2/29/2003"	false
2	The method shall return false for the 31st on any month that ends on the 30th	Create an instance of Date with April 31 **Test input:** "4/31/2003"	false
3	The method shall return false for an invalid month (1-12)	Create an instance of Date with a 13th month **Test input:** "13/31/2003"	false
4	The method shall return false for an invalid date	Create an instance of Date with a date greater than 31 **Test input:** "3/32/2003"	false
5	The method shall return false for a negative value	Create an instance of Date with a negative number as the month **Test input:** "-1/31/2003"	false
6	The method shall return true for valid dates	Create an instance of Date with September 2, 2022 **Test input:** "9/2/2022"	true
7	The method shall return true for valid dates	Create an instance of Date with December 20, 2004 **Test input:** "12/20/2004"	true

class name: Student
method signature: public int compareTo(Student std) {} // compare two Student objects

Test Case #	Requirement	Test description and Input Data	Expected result/output
1	The method shall return a negative integer if this last name comes before the inputted last name lexicographically.	Create two instances of student with last name "Castellanos"(s2) and "Vergara"(s1) and compare the former to the latter **Test input:** s2.compareTo(s1)	negative integer
2	The method shall return a positive integer if this last name comes after the inputted last name lexicographically.	Create two instances of student with last name "Castellanos"(s2) and "Vergara"(s1) and compare the latter to the former **Test input:** s1.compareTo(s2)	positive integer
3	The method shall return a negative integer if these last names are the same, but first name comes before the inputted first name lexicographically.	Create two instances of student with same last names, but first names "Luis"(s3) and "Marlon"(s1) and compare the former to the latter **Test input:** s3.compareTo(s1)	negative integer
4	The method shall return a positive integer if these last names are the same, but first name comes after the inputted first name lexicographically.	Create two instances of student with same last names, but first names "Luis"(s3) and "Marlon"(s1) and compare the latter to the former **Test input:** s1.compareTo(s3)	positive integer
5	The method shall return a negative integer if these full names are the same, but this date of birth comes before the inputted date of birth.	Create two instances of student with same full names, but date of births are 4/13/2002(s2) and 4/12/2002(s1) and compare the former to the latter **Test input:** s2.compareTo(s1)	negative integer
6	The method shall return a positive integer if these full names are the same, but this date of birth comes after the inputted date of birth.	Create two instances of student with same full names, but date of births are 4/13/2002(s2) and 4/12/2002(s1) and compare the latter to the former **Test input:** s1.compareTo(s2)	positive integer
7	The method shall return 0 if both profiles are the same completely.	Create two instances of student with same full names and dates of birth and compare them **Tests input:** s1.compareTo(s2)	0