|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: Date  Method Signature: public Boolean isValid() | | | |
| Test Case # | Requirement | Test Description and Input Data | Expected output |
| 1 | The method should return “Invalid” if non-leap year February has 29 as input days. | Create an instance of Date with non-leap year and let month be February and set days as 29.  Test Input: 2/29/2025 | Invalid |
| 2 | The method should return “Invalid” if the input days is greater than 30 for month of April, June, September, November. | Create an instance of Date with valid year, set days as 31, and let month be November.  Test Input: 11/31/2024 | Invalid |
| 3 | Valid range for days should be greater than zero. | Create an instance of Date with valid year and month, and let days be -5.  Test Input: 10/-6/2024 | Invalid |
| 4 | Valid range for the month should be between 1 and 12. | Create an instance of Date with valid year and days, and let month be 14.  Test Input: 14/5/2024 | Invalid |
| 5 | The method should return “Valid” for a valid leap year date. | Create an instance of Date with leap year and let month be February and set days as 29.  Test Input: 2/29/2024 | Valid |
| 6 | The method should return “Valid” for correct date. | Create an instance of Date with any standard date that you see in calendar, example today.  Test Input: 9/30/2024 | Valid |

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: Profile  Method Signature: public int compareTo(Profile other) | | | |
| Test Case # | Requirement | Test Description and Input Data | Expected output |
| 1 | Compare by last name | Comparing two profiles where the first profile last name comes alphabetically before the second profile last name.  Test Input: Profile("Charlie", "Brown", new Date(88, 11, 30)).compareTo(Profile("Alice", "Smith", new Date(95, 4, 15))) | -1 |
| 2 | Compare by first name | Comparing two profiles where the first profile first name comes alphabetically before the second profile first name. And both last names are same.  Test Input: Profile("Alice", "Smith", new Date(95, 4, 15)).compareTo(Profile("Bob", "Smith", new Date(90, 7, 20))) | -1 |
| 3 | Compare by date of birth | Comparing two profiles where the first profile date of birth comes before the second profile date of birth. And has same last and first names.  Test Input: Profile("Alice", "Smith", new Date(92, 1, 10)).compareTo(Profile("Alice", "Smith", new Date(95, 4, 15))) | -1 |
| 4 | Compare by last name | Comparing two profiles where the first profile last name comes alphabetically after the second profile last name.  Test Input: Profile("Alice", "Smith", new Date(95, 4, 15)).compareTo(Profile("Charlie", "Brown", new Date(88, 11, 30))) | 1 |
| 5 | Compare by first name | Comparing two profiles where the first profile first name comes alphabetically after the second profile first name. And both last names are same.  Test Input: Profile("Bob", "Smith", new Date(90, 7, 20)).compareTo(Profile("Alice", "Smith", new Date(95, 4, 15))) | 1 |
| 6 | Compare by date of birth | Comparing two profiles where the first profile date of birth comes after the second profile date of birth. And has same last and first names.  Test Input: Profile("Alice", "Smith", new Date(95, 4, 15)).compareTo(Profile("Alice", "Smith", new Date(92, 1, 10))) | 1 |
| 7 | Duplicate Profile | Comparing two profiles which are the same, date of birth, first and last name.  Test Input: Profile("Alice", "Smith", new Date(95, 4, 15)).compareTo(Profile("Alice", "Smith", new Date(95, 4, 15))) | 0 |