In this problem, you will create a class Date (which represents a year, month, day) and its subclass DateTime which includes an hour and minute of the day.

While most of the attributes are integers, it will store the month as a 3-letter abbreviation (e.g. 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', or 'Dec'). Remember that 'Apr', 'Jun', 'Sep', and 'Nov' have 30 days, 'Feb' has either 28 or 29, and all others have 31 days. The attributes of the two classes are as follows:

Date

Attribute	Invariant	Category
MONTHS	list of 3-letter month abbreviations in order	Class attribute
DAYS	dictionary from months to number of days	Class attribute
_year	int >= 2000	Immutable instance attribute
_month	3-letter string abbreviation	Immutable instance attribute
_day	int that is a valid day of _month	Mutable instance attribute

DateTime (in addition to those inherited)

Attribute	Invariant	Category
_hour	int in range 0..23	Mutable instance attribute
_minute	int in range 0..59	Mutable instance attribute

Instructions:
1. Fill in the missing information in each class header.
2. Add any necessary class attributes
3. Add getters and setters as appropriate for the instance attributes
4. Fill in the parameters of each method (beyond the getters and setters)
5. Implement each method according to the specification.
6. Enforce any preconditions in these methods using asserts

Headers for the getters and setters have not been added. You are to write these from scratch (need not write specifications for them). For the other methods, pay attention to the provided specifications. The only parameters are those in the preconditions. The class DateTime may not use any attribute or getter/setter inherited from Date. It may only use super() to access overridden methods.

Enforce preconditions with assert unless you are given a specific error to use instead. Type-based preconditions should all be managed with isinstance and not type.

Finally, there is the matter of February. In the DAYS class attribute, you should consider February as having 28 days, and ignore leap years. However, you should not ignore leap years (February has 29 days) when enforcing the invariant of the _day attribute. To help you with that use the following helper function (to be implemented by you).

```
def isleapyear(y):
"""Returns True if y is a leap year. False otherwise
Precondition: y is an int >= 0"""
```

(a) **The class Date**

```
class Date(): # Fill in missing part
    """A class representing a month, day and year
    Attribute MONTHS: A CLASS ATTRIBUTE list of all month abbreviations
    in order
    Attribute DAYS: A CLASS ATTRIBUTE that is a dictionary. Keys are the
    strings from MONTHS; values are days in that month ('Feb' is 28 days)"""
    # Attribute _year: The represented year. An int >= 2000 (IMMUTABLE)
    # Attribute _month: The month. A valid 3-letter string from MONTHS
    (IMMUTABLE)
    # Attribute _day: The day. An int representing a valid day of _month
    (MUTABLE)

    # CLASS ATTRIBUTES. ( Fill in missing part)
    'Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'
    'Jan':31,'Feb':28,'Mar':31,'Apr':30,'May':31,'Jun':30,'Jul':31,'Aug':31,
    'Sep':30,'Oct':31,'Nov':30,'Dec':31

    # DEFINE GETTERS/SETTERS/HELPERS AS APPROPRIATE. SPECIFICATIONS
    NOT NEEDED.

    def
        """Returns the year of this date"""
        # Fill in missing part
```

```
def
        """Returns the month of this date"""
        # Fill in missing part

def
        """Returns the day of this date"""
        # Fill in missing part

def
        """Sets the day of this date
        Parameter value: The new day
        Precondition: value is a valid day in the month"""
        # Fill in missing part

def __init__(): # Fill in missing part
        """Initializes a new date for the given month, day, and year

        Precondition: y is an int >= 2000 for the year
        Precondition: m is a 3-letter string for a valid month

        Precondition: d is an int and a valid day for month m"""
        # Fill in missing part


def __str__(): # Fill in missing part
        """Returns a string representation of this date.
        The representation is month day, year like this: 'Jan 2, 2002' """
        # Fill in missing part

def __lt__(): # Fill in missing part
        """Returns True if this date happened before other (False otherwise)

        Precondition: other is a Date

        This method causes a TypeError if the precondition is violated."""

# IMPORTANT: You are limited to 20 lines. Do NOT brute force this.
```

```
        # Fill in missing part
```

(b) The class DateTime.

```
class DateTime():  # Fill in missing part
        """A class representing a month, day and year, plus time of day (hours,
        minutes)"""
        # Attribute _hour: The hour of the day. An int in range 0..23 (MUTABLE)
        # Attribute _minute: The minute of the hour. An int in range 0..59
        (MUTABLE)

        # DEFINE GETTERS/SETTERS/HELPERS AS APPROPRIATE.
        SPECIFICATIONS NOT NEEDED.

        def
                """Returns the hour of the day"""
                # Fill in missing part

        def
                """Sets the hour of the day
                Parameter value: The new hour
                Precondition: hour is an int in 0..23"""
                # Fill in missing part

        def
                """Returns the minute of the hour"""
                # Fill in missing part

        def
                """Sets the hour of the day
                Parameter value: The new hour
                Precondition: hour is an int in 0..23"""
                # Fill in missing part

        def __init__(): # Fill in missing part
                """Initializes a new datetime for the given month, day, year, hour and minute
        This method adds two additional (default) parameters to the initialize for
        Date. They are hr (for hour) and mn (for minute).
        Precondition: y is an int >= 2000 for the year
        Precondition: m is a 3-letter string for a valid month
        Precondition: d is an int and a valid day for month m
        Precondition: hr is an int in the range 0..23 (OPTIONAL; default 0)
        Precondition: mn is an int in the range 0..59 (OPTIONAL; default 0)"""
        # Fill in missing part


    def __str__(): # Fill in missing part
        """Returns a string representation of this DateTime object
        The representation is 'hh:mm on month day, year' like this: '9:45 on Jan 2, 2002'
        Single digit minutes should be padded with 0s. Hours do not need to be
        padded."""
        # Fill in missing part
```