Submission: See course shell for how and when to submit

Creating a Date class [Marks : 40]

We will assume that there are 12 months in a year and each month is 30 days long. In this course, sometimes it is more practical to ignore certain details, that would otherwise overly complicate your model.

Date Class
Fields
- year : int - month : int - day : int
Methods
+ «constructor» Date(day : int = 1, month : int = 1, year : int = 2022) + ToString() : string + Add(days : int) : void + Add(month : int, days : int) : void + Add(other : Date) : void - Normalize() : void

Description of field members:

1. **year**: this private member represents the year value of this object
2. **month**: this private member represents the month value of this object. You may interpret the months as you normally do i.e. 1 = January, 2 = February
3. **day**: this private member represents the day value of this object

> Alternately you may choose to have 0 = January, 1 = February etc.. This would be my personal preference.

Description of constructor:

1. **`public Date(int day, int month, int year)`**: this is the constructor of this class. It takes three integer arguments and assigns them to the appropriate fields. You will assume that all the arguments will be in the correct range.

Description of action members:

1. **`public override string ToString()`**: this method overrides the same method in the base class. It does not take any argument but it returns a string representing this object.
2. **`public void Add(int days)`**: this method takes a single integer argument representing the amount to increase the day field by and adds it to the appropriate field.
3. **`public void Add(int days, int months)`**: this method takes two integer arguments representing the amount to increase the month field and the day field. It adds the argument to the appropriate fields.
4. **`public void Add(Date other)`**: this method takes a single integer argument representing a date. The is actually three values packed into a single date object. You will have to separate each packed value and increase the year, month and day fields by the appropriate amount.
5. **`private void Normalize()`**: this method does not take any argument nor does it return a value. It does the following:
   a. If the days field is more than the number of days in a month, then it subtracts the number of days in the month from the day field and increases the month field by one.
   b. If the month field is more than 12 then, 12 is subtracted from the month field and the year field is increased by 1.

   This method should be called at the end of the constructor, and each Add method.

In your main method write the code to do the following:

1. Create a Date object. You decide on the arguments.
2. Display the above object reference on the console.
3. Call the **`Add()`** method of the object reference to verify that the method works correctly. You will need to invoke this method three time with different number and types of argument because this method is overloaded three times. (How would you display the modified object?).
4. What would you add to the class to make the display more sensible? i.e. the day and month value should be in range

Enhancements:

1. Display the text for the month i.e. Jan or January for the month 1.
2. Handle the different month length properly
   "**30 days** has September, April, June, and November. All the rest have **31**. Save February, with **28 days** clear, and **29** each leap year."
3. Would it be better to have a single int field that represents the number of days passed since the year has started? Now all the operations are easier to implement excepting the ToString method.