# 1. Calendar

A calendar allows users to create TODO events on certain dates.
Let's create a simple calendar application by implementing two classes, `MyDate` and `MyCalendar`.

## 1.1 MyDate

Create a class `MyDate` based on the following requirements.

**Fields**

```
private int year;
private int month;
private int day;
```

**Constructor**

```
public MyDate(int year, int month, int day)
```

We can assume that `year`, `month` and `day` will always be valid.

**Methods**

```
public void addDays(int days)
```

Add `days` to the current date, which will then become the new date after calculation.

```
public String toString()
```

Return a string of the formatted date (yyyy-mm-dd), like 2023-01-31.

```
public static int difference(MyDate date1, MyDate date2)
```

Return the difference (number of days) between `date1` and `date2`.
In particular, if `date1` is earlier than `date2`,
the return value should be <0; Otherwise the return value should be >= 0.

## 1.2 MyCalendar

Create a class `MyCalendar`, which has the following methods to support two features.

**Constructors**

```
public MyCalendar(int capacity)
```

Create a new calendar with the maximum number of events specified by `capacity`, i.e., the number of events in this calendar cannot exceed `capacity`.

**Methods**

```
public boolean addEvent(MyDate date, String eventName)
```

This method adds an event with name `eventName` to a given `date`.
If the calendar's capacity has been exceeded, return `false`; otherwise return `true`.

```
public String finishNextEvent()
```

A calendar may have several events.
Whenever we call `finishNextEvent()`, it means that we finish the **next** event, by **chronological order** (i.e., the events will be finished one by one from the earliest to the latest).
The return value of this method is the name of the **next** event.

For example, the first time `finishNextEvent()` is called, it returns the earliest event's name; the second time `finishNextEvent()` is called, it returns the second-earliest event's name, and so on.

If there are multiple events on the same date,
these events should be returned by their **alphabetical order** (lexicographically).
For example, "laundry" should be returned before "meeting", because
"laundry" precedes "meeting" in alphabetical order (You may check the `String.compareTo` method for comparing strings by alphabetical order).

If all events have already been returned, this method should return `"NONE"`.

If an event has already been finished, it doesn't count for `capacity`.

**Fields (and other methods)**

You could define any fields and other methods as you want to help you implement the requirements.
Our tests will only invoke the required methods to evaluate the correctness of your code.