Problem Description

Design, Implement a class that will calculate the "cumulative return" for given base date and an as of date based on historical daily returns.

The cumulative return for a specific ***as of date A*** with ***base date B*** is:

> ***CR = [(1+DR$_1$) * (1+DR$_2$) * (1+DR$_3$) * ...] - 1.0***, where
>
> CR = Cumulative Return
>
> DR$_1$, DR$_2$, DR$_3$, ... are all daily returns from B through A (***excluding B and including A***).

Inputs

- Map<Date, Double> dailyReturns : Daily returns for various dates from B
- Date baseDate : Base date typically means that we don NOT have any daily return data before this date
- Date asOfDate : Date for which user needs cumulative return

Implementation Specification

Solution Template

```
// IMPLEMENT THIS CLASS AND ITS METHODS.
// Please document your assumptions, design decisions
// as inline comments or in a separate text file.

class CumRetCalulator {
    // Use any data structure for storage as you see fit

    public CumRetCalcualtor(Map<Date, Double> dailyReturns) {
        // This is the constructor which loads daily return data.
        // In a practical scenario, we might get historical daily return data from outside
        // vendors for 5, 10, 20 or even 30 years. So, considering 260 working days in a year,
        // for 30 years, there can be data in the range of high-thousands (~10,000 rows) line items.
        // Considering this big size, ideally, it should be an one-time operation (think about Singlet

    }

    double findCumReturn(Date asof, Date base) {
        // Calculate cumulative return for 'asof' date from 'base' date.
        // See examples of calculation below
        // Keep in mind that this method can be invoked for many number of
        // as of dates (1000s) in a short span of time.

    }
}
```

Example

Assuming this is the dailyReturns map

Date	**Daily return (in decimal)**
2015-01-10	0.10 (means 10% daily return on 10th Jan 2015)
2015-02-10	0.05
2015-04-10	0.15
2015-04-15	-0.10
2015-06-10	-0.12

Then cumulative returns for following dates, using **base date of 2015-02-01** are:

As of Date	**Cumulative Return**	**Remarks**
2015-01-31	null	As of Date is before Base Date
2015-02-28	0.05	(1+0.05) - 1
2015-03-13	0.05	Same as above
2015-04-30	0.08675	(1+0.05) * (1+0.15) * (1-0.10) - 1
2015-05-08	0.08675	Same as above
2015-06-30	-0.04366	(1+0.05) * (1+0.15) * (1-0.10) * (1-0.12) - 1

Efficiency

Please think about time complexity. Though brute force is acceptable, please document your thoughts on how to improve in case you're going for brute force solution. Remember, there can be many solutions and no "one cure-all" solution. We would like to see the design decisions you make.