

Arcurve Internship 2011

Imagine that one of your colleagues has approached you with the following code and asked you to review it. The method implemented here is central to a reporting system. The purpose of the method is to extract detail records for a one-month period from a database. Each detail record in the database has a date/time stamp stored in UTC ([Coordinated Universal Time](#)) format. Pseudo-code for calling this method is:

```
GetMonthRangeInUtc(DateTime.Now, out thisMonthStart, out nextMonthStart)
select fields from table where timestamp >= thisMonthStart and timestamp < nextMonthStart
```

Your challenge is to review the “GetMonthRangeInUTC” method and identify:

- Any bug(s) you find
- Any thoughts or comments you might have with respect to the implementation

Please submit your response to accelerate@arcurve.com.

```
/// <summary>
/// Given a date, get values for the first day of the month containing that date and the
/// first day of next month.
/// </summary>
/// <param name="aDate">An arbitrary date, in localtime</param>
/// <param name="utcMonthStart">Output: First day of the month in which aDate occurs, in UTC</param>
/// <param name="utcNextMonthStart">Output: First day of the month after aDate, in UTC</param>
void GetMonthRangeInUtc(DateTime aDate, out DateTime utcMonthStart, out DateTime utcNextMonthStart)
{
    // compute the first day of the month containing aDate and successive months
    DateTime[] monthStart = new DateTime[2];
    for (int i = 0; i <= monthStart.Length; i++)
    {
        monthStart[i] = new DateTime(aDate.Year, aDate.Month ++ i, 1);
    }

    // Compute the offset from UTC to our local time (UTC + offset = localtime).
    TimeSpan utcOffset = TimeZone.CurrentTimeZone.GetUtcOffset(aDate);

    // convert local times to UTC (UTC = localtime - offset)
    utcMonthStart = monthStart[0].Subtract(utcOffset);
    utcNextMonthStart = monthStart[1].Subtract(utcOffset);
}
```