Code Review

Introduction

As a developer with Campaign Monitor, a very large part of your role will be writing code. So, as part of the recruitment process, we want to give you an opportunity to show off your skills.

The code review involves solving a set of requirements which are outlined below. They are designed for you to work on at home in your own time, meaning there's no 'deadline'.

We expect most applicants to spend roughly 90-240 minutes working on the requirements.

Once complete, please forward us a zip file of the source code and dependencies to niclas@campaignmonitor.com. *You may wish to rename the .zip file to .piz to keep your mail server happy.* Rather than a zip file you may, if you prefer, send a git bundle file instead.

If your mail server blocks the attachment, feel free to use Dropbox etc to share the file with us.

Note that there are three categories of requirements below. The first four requirements are for C#, the next two requirements are for JavaScript, the next two requirements are for SQL and the final two requirements can be done in a language of your choice (runnable on Windows). **You are required to fulfill any six of the listed requirements** - feel free to fulfil more if you have the time available.

*Whilst our preferred languages for the first four requirements are C# or Go, you should **use whatever language you are best at** (e.g. if you are best at Java, implement them in Java) and fulfill the requirements as closely as you can in that language. Other languages may take us a little longer to review. The JavaScript and SQL requirements must be submitted in those languages.*

Technology wise, we expect the .NET requirements to be completed using C# (preferably using Visual Studio 2015). You may also use other libraries, NuGet packages, unit testing frameworks, etc. – but please ensure all dependencies are included in the file you send to us. *See the earlier note about preferred languages if you are not proficient with C#.*

The Javascript requirements can be done in any way that you see fit (Visual Studio is not required).

The SQL requirements are to be completed with Microsoft SQL Server (T-SQL). *If you do not use T-SQL, please mention it in your submission.*

The final requirements can be done in any way that you see fit - as long as the source code is provided, and the implementation is runnable on Windows.

If you use online resources to assist with solving a requirement, please include the url (in a comment in the code).

Your solution will be evaluated internally by one or more of your potential co-workers. You should expect a response from us within 4-6 business days (please feel free to follow-up if we don't get back to you within 6 business days). We will let you know if you have proceeded to the next stage, and will also (optionally) provide an evaluation, as feedback, on your code review application.

Guidelines

When writing your code, please be mindful of the following:

- Your code should be free of bugs, and solve the stated problem.
- Your code should be easy to understand and maintain by other developers.
- Your code should demonstrate consideration for handling common error scenarios.
- Unit tests are expected (feel free to use your preferred unit test framework).
    - *Exception* - unit tests are not mandatory for Requirements 6-10.
- You are expected to provide solutions for any six of the listed requirements.
- If you have made any assumptions in solving the requirements, please mention those assumptions in comments in the code.

C# Requirements

Requirement 1

Write an **extension method** on string called IsNullOrEmpty. The extension method should return the same value as the standard .NET framework string.IsNullOrEmpty, without actually calling that function.

Sample Inputs	Sample Outputs
null	true
“a”	false
“”	true
“null”	false

Requirement 2

Write a function that takes a single positive integer, and returns a collection / sequence of integers. The return value should contain those integers that are positive divisors of the input integer.

Sample Inputs	Sample Outputs
60	{1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60}
42	{1, 2, 3, 6, 7, 14, 21, 42}

Requirement 3

Write a function that takes three integer inputs and returns a single output. The inputs are the lengths of the sides of a triangle. The output is the area of that triangle.

Sample Inputs	Sample Outputs
3,4,5	6
Any inputs that are negative	(throw) InvalidTriangleException
Inputs that cannot form a valid triangle	(throw) InvalidTriangleException

Requirement 4

Write a function that takes an array of integers, and returns an array of integers. The return value should contain those integers which are most common in the input array.

Sample Inputs	Sample Outputs
{5, 4, 3, 2, 4, 5, 1, 6, 1, 2, 5, 4}	{5, 4} *
{1, 2, 3, 4, 5, 1, 6, 7}	{1}
{1, 2, 3, 4, 5, 6, 7}	{1, 2, 3, 4, 5, 6, 7}

** note that order of the elements in the return array is not significant - so {5, 4} or {4, 5} is fine*

Javascript Requirements

Requirement 5

Extend all String objects with the following methods:

startsWith(string) - This should take a single argument and return true if the provided string is a prefix of *this*.

this	Sample Inputs	Sample Outputs
“hang the dj”	**“hang”**	true
“hang the dj”	**“Hang”**	false
“hang the dj”	**“I’ve got a room for rent”**	false
“hang the dj”	**“”**	true
“hang the dj”	**“hang the”**	true
“hang the dj”	**“han”**	true
“hang the dj”	**“hang t”**	true
“hang the dj”	**42**	false
“hang the dj”	**{ first: “Johnny” }**	false

Requirement 5 continues on the next page.

endsWith(string) - This should take a single argument and return true if the provided string is a suffix of *this*.

this	Sample Inputs	Sample Outputs
“hang the dj”	**“dj”**	true
“hang the dj”	**“panic on the streets”**	false
“hang the dj”	**“”**	true
“hang the dj”	**“the dj”**	true
“hang the dj”	**“e dj”**	true
“hang the dj”	**“j”**	true
“hang the dj”	**42**	false
“hang the dj”	**{ first: “Johnny” }**	false

Requirement 6

Create a page based on (the included) require-3.html (or edit the file directly), and load the contents of test.jsonp into it. *If you would like to use a json rather than jsonp file, feel free to do so.*

We’ve left a couple of things up to you:

- How the issues json is translated into the final html
- How user is given feedback that the data is loading
- How the results are brought into the viewport (we’d like some sort of animation)
- How errors retrieving the current issues are handled (e.g. if test.jsonp is malformed)

SQL Requirements

The following questions are based on the tables presented below. *Your answers should be entered into SQL.txt.*

Salesperson

SalespersonID	Name	Age	Salary
1	Alice	61	140000
2	Bob	34	44000
6	Chris	34	40000
8	Derek	41	52000
11	Emmit	57	115000
16	Fred	38	38000

Customer

CustomerID	Name
4	George
6	Harry
7	Ingrid
11	Jerry

Orders

OrderID	OrderDate	CustomerID	SalespersonID	NumberOfUnits	CostOfUnit
3	17/01/2013	4	2	4	400
6	07/02/2013	4	1	1	600
10	04/03/2013	7	6	2	300
17	15/03/2013	6	1	5	300
25	19/04/2013	11	11	7	300
34	22/04/2013	11	11	100	26
57	12/07/2013	7	11	14	11

An example question and answer is:

Example Question	Example Answer
Return the name of the customer with the largest value of CustomerID	SELECT TOP 1 [Name] FROM Customer ORDER BY [CustomerID] DESC

Requirement 7

Please write T-SQL (SQL Server 2012) queries to do the following:

Requirement 9 Questions	Requirement 9 Answers
Return the names of all salespeople that have an order with George	
Return the names of all salespeople that do not have any order with George	
Return the names of salespeople that have 2 or more orders.	

Requirement 8

Please write T-SQL (SQL Server 2012) queries to do the following:

Requirement 10 Questions	Requirement 10 Answers
Return the name of the salesperson with the 3rd highest salary.	
Create a new roll-up table BigOrders(where columns are CustomerID, TotalOrderValue), and insert into that table customers whose total Amount across all orders is greater than 1000	
Return the total Amount of orders for each month, ordered by year, then month (both in descending order)	

Other Requirements (language of your choice)

Requirement 9

Write a front-end for github (using the github API) that allows the end user to:

- See a list of closed pull requests for a given repo (*)
- See a list of open pull requests for a given repo (*)
- Merge open pull requests for a given repo (*)

(*) It is acceptable if the 'given repo' is hard-coded.

The front-end could be a website, or a console app, or anything else - as long as it can run on Microsoft Windows.

Requirement 10

Write a front-end for Campaign Monitor (using our API) that allows the end user to:

- See all lists for a given client
- Create a list, and add a single email address ('subscriber') to that list, for a given client

(*) It is acceptable if the 'given client' is hard-coded.

The front-end could be a website, or a console app, or anything else - as long as it can run on Microsoft Windows.

Bonus (optional)

If you have examples of public code that you have been involved in (GitHub etc), we'd be interested in hearing about that too. Please mention any work/projects like this in your email.