# **Coding Assessment**

## Paid Development Opportunity with Prof. Mohamed Ali

Prerequisites

This assignment is designed to evaluate your technical, analytical and problem-solving skills. The entire assignment needs to be completed using Microsoft Visual Studio.

Link for downloading a free version <https://visualstudio.microsoft.com/downloads/>

Here Community version can be downloaded for free.

You might need to add [Microsoft.SqlServer.Types](https://www.nuget.org/packages/Microsoft.SqlServer.Types/) dll to your project. You can come up with your implementation of *distance calculation* in case you don’t want/unable to use the the dll in .NET framework.

# Problems

1. Perform the following steps:
   1. Create a class *Coordinates* will two data members, *lat* and *lng*, representing latitude and longitude for a coordinate.
   2. Generate a static function *GenerateRandCor* taking a *cor* (Coordinate), *radius* (double) and *n* (int) as input parameters. The objective of the function is to generate and return *n* random coordinates within the circle with centre as *cor* and radius *radius* in kilometres*.* Any two coordinates within the output list should have a minimum distance of 400 meters.

Useful resources:

[C# SqlGeography implementation](https://docs.microsoft.com/en-us/dotnet/api/microsoft.sqlserver.types.sqlgeography.stgeomfromtext?view=sql-smo-140.17283.0)

[C# SqlGeography.STDistance documentation](https://docs.microsoft.com/en-us/dotnet/api/microsoft.sqlserver.types.sqlgeography.stdistance?view=sql-smo-140.17283.0)

[C# SqlGeography.Point documentation](https://docs.microsoft.com/en-us/dotnet/api/microsoft.sqlserver.types.sqlgeography.point?view=sql-smo-140.17283.0)

[Stackoverflow - convert-sql-geography-to-c-sharp](https://stackoverflow.com/questions/16456365/convert-sql-geography-to-c-sharp)

* 1. Determine the time complexity of the above algorithm.
  2. Is it possible to generate a similar algorithm having time complexity of *O(n)*?

Submit results for these parameters: *cor*: 47.260705, -122.482980, *radius*: 4km, *n*: 20

Hint: to solve part d, you can take inspiration from [two-sum-problem](https://leetcode.com/problems/two-sum/) *O(n)* implantation.

1. Perform the following tasks:
   1. Create a class *Route* with the following data members:
      * distance (double)
      * time (double)
      * maneuverPoints (List<*Coordinates*>)
   2. Generate a static function *GenerateRequestUrl* which takes two *Coordinates* object as input and returns Bing Routing API request URL (String). You can generate your *BingMapsKey* using a free student/developer account. The function should pass the two input parameters as starting and end of the route in the request URL. You can free to use other routing parameters if required.

Submit results for these parameters: toC*or*: 47.260705, -122.482980, *fromCor*: 47.244913, -122.438933

Useful resources:

[Bing API Key](https://docs.microsoft.com/en-us/bingmaps/getting-started/bing-maps-dev-center-help/getting-a-bing-maps-key)

[Bing Routing Documentation](https://docs.microsoft.com/en-us/bingmaps/rest-services/routes/)

[Driving Route Example](https://docs.microsoft.com/en-us/bingmaps/rest-services/examples/driving-route-example)

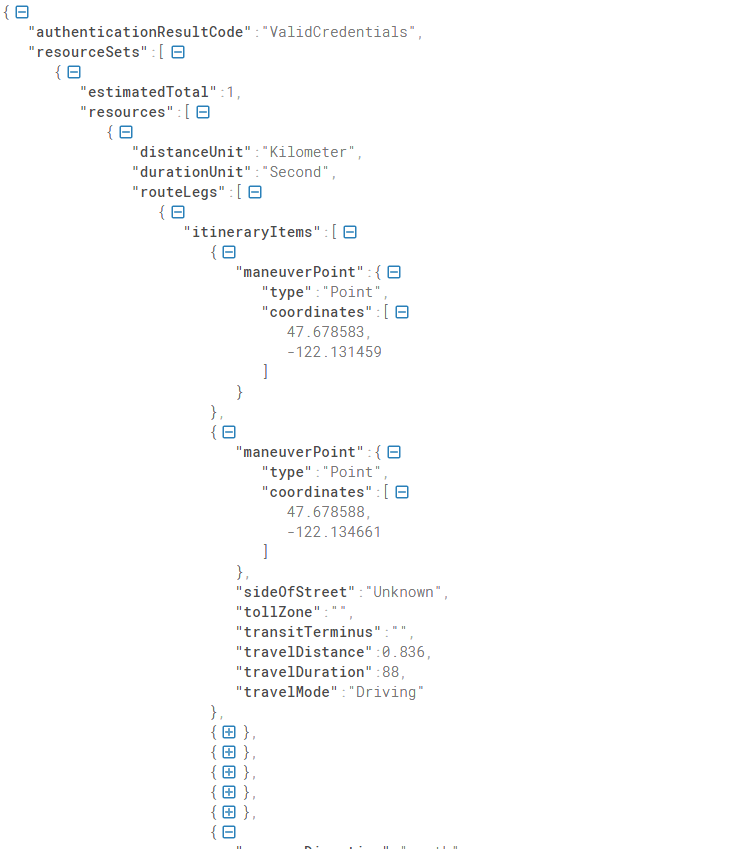
* 1. Declare another static function *GetJsonResponse* which takes the request URL as input and returns the Routing API response JSON string. The response should look like the response in the [example](https://docs.microsoft.com/en-us/bingmaps/rest-services/examples/driving-route-example). You can choose to use JSON or XML as output (Recommended JSON).

Useful resources:

<https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/console-webapiclient>

<https://stackoverflow.com/questions/10226089/restsharp-simple-complete-example>

* 1. Extract the useful information from the JSON (or XML) response and instantiate the class *Route*. Use this structure the extract information.
     + distance = resourceSets -> resources -> travelDistance
     + time = resourceSets -> resources -> travelDuration
     + maneuverPoints = resourceSets -> resources -> routeLegs -> itineraryItems (extract all coordinates from maneuverPoint -> coordinates from each itineraryItems)



Useful Resources:

[JSON Formatter](https://jsonformatter.curiousconcept.com/)

1. Show the results of instantiated *Route* to front end using simple HTML structure. You can add CSS to show your proficiency in front-end development. In case you are not able to complete the previous question, use dummy data to instantiate the *Route* class*.* The objective of this problem is to complete a front-end page in ASP.NET framework while establishing front-back connectivity.

Hint: First create an ASP.NET MVC project, then create a new view and controller for your front-end page. The objective is to retrieve view data from the controller and displaying in a table on the HTML page. You can have any elements on the HTML page as it suits the application.

Deliverables:

1. Complete ASP.NET project solution as a zip folder/git link.
2. Sample results from question 1 and 2.
3. Screenshots from question 3.