The test will be done using Coderpad (www.coderpad.io)
There's an embedded Sandbox where you can try your code here
https://coderpad.io/resources/docs/getting-started-with-coderpad/candidates/

You will be able to Google for hints during the interview.

Context:

Given the following:

List of Events

- Event object example:

```
 var events = new List<Event>{
    new Event{ Name = "Phantom of the Opera", City = "New York"},
    new Event{ Name = "Metallica", City = "Los Angeles"},
    new Event{ Name = "Metallica", City = "New York"},
    new Event{ Name = "Metallica", City = "Boston"},
    new Event{ Name = "LadyGaGa", City = "New York"},
    new Event{ Name = "LadyGaGa", City = "Boston"},
    new Event{ Name = "LadyGaGa", City = "Chicago"},
    new Event{ Name = "LadyGaGa", City = "San Francisco"},
    new Event{ Name = "LadyGaGa", City = "Washington"}
 };
```

- Customer

```
var customer = new Customer{ Name = "Mr. Fake", City = "New York"};
```

- Methods:

  GetDistance(string fromCity, string toCity)
  AddToEmail(Customer, Event)
  and a few others i didnt have to use and i dont remember the details

Imagine we're running a business where doing ticket selling.
So we have a website where we are selling tickets for live events. The specific business scenario here is that we want to do an email campaign.
So imagine where building a service (component) for this email campaign what we want to do is to compose an email with the list of all the live events and send it to the customer.
In the main function, you'll find some test data with a list of events and a single customer.
You don't have to worry about the sending part as it's covered by the function that you'll find in the code.
You'll just need to figure out what are the events that we want to send to the customer that the customer may be interested in.

Questions:
1. Imagine this is the interface or an API, so you don't care how he works. You just need to know that you need to call this function to add an event to the email that you want to send to the customer. The function takes two parameters as the input: the customer and the event. So the

first task here is to prepare the email for this customer in the main function to send all the events.

Write a code to add all events in the customer's location to the email. Considering the objects shared above:

1. What should be your approach to getting the list of events?
2. How would you call the **AddToEmail** method in order to send the events in an email?
3. What is the expected output if we only have the client John Smith?
4. Do you believe there is a way to improve the code you first wrote?

2. We want to send the events occurring in the client’s city, but also the ones that occur in the nearest city. For this, there’s another function called **GetDistance**. This function takes two inputs: from city and to city to calculate the distance. If you put from city and to city as the same city, the distance will be 0 (zero). You will need to send 5 events to the customer, so you need to get the closest to them.

Write a code to add the 5 closest events to the customer's location to the email.

1. What should be your approach to getting the distance between the customer’s city and the other cities on the list?
2. How would you get the 5 closest events and how would you send them to the client in an email?
3. What is the expected output if we only have the client John Smith?
4. Do you believe there is a way to improve the code you first wrote?

3. If the GetDistance method is an API call which could fail or is too expensive, how will u improve the code written in 2? Write the code.
4. If the GetDistance method can fail, we don't want the process to fail. What can be done? Code it. (Ask clarifying questions to be clear about what is expected business-wise)
5. If we also want to sort the resulting events by other fields like price, etc. to determine which ones to send to the customer, how would you implement it? Code it.

One of the questions is: how do you verify that what you’ve done is correct.

Lamba Expressions. Data Structures. Dictionary, ContainsKey method

PLEASE SEE THE CODEBASE BELOW

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
namespace Viagogo {
   public class Event
```

```
{
    public string Name{ get; set; }
    public string City{ get; set; }
}
public class Customer
{
    public string Name{ get; set; }
    public string City{ get; set; }
}
public class Solution
{
    static void Main(string[] args)
    {
        var events = new List<Event>{
            new Event{ Name = "Phantom of the Opera", City = "New York"},
            new Event{ Name = "Metallica", City = "Los Angeles"},
            new Event{ Name = "Metallica", City = "New York"},
            new Event{ Name = "Metallica", City = "Boston"},
            new Event{ Name = "LadyGaGa", City = "New York"},
            new Event{ Name = "LadyGaGa", City = "Boston"},
            new Event{ Name = "LadyGaGa", City = "Chicago"},
            new Event{ Name = "LadyGaGa", City = "San Francisco"},
            new Event{ Name = "LadyGaGa", City = "Washington"}
        };
        //1. find out all events that arein cities of customer
        // then add to email.
        var customer = new Customer{ Name = "Mr. Fake", City = "New York"};
        var query = from result in customer
                    where result.Contains("New York")
                    select result;
        // 1. TASK
        foreach(var item in events)
        {
            AddToEmail(query, item);
        }
        /*
         * We want you to send an email to this customer with all events in their city
         * Just call AddToEmail(customer, event) for each event you think they should get
         */
    }
    // You do not need to know how these methods work
    static void AddToEmail(Customer c, Event e, int? price = null)
```

```
        {
            var distance = GetDistance(c.City, e.City);
            Console.Out.WriteLine($"{c.Name}: {e.Name} in {e.City}"
                + (distance > 0 ? $" ({distance} miles away)" : "")
                + (price.HasValue ? $" for ${price}" : ""));
        }
        static int GetPrice(Event e)
        {
            return (AlphebiticalDistance(e.City, "") + AlphebiticalDistance(e.Name, "")) / 10;
        }
        static int GetDistance(string fromCity, string toCity)
        {
            return AlphebiticalDistance(fromCity, toCity);
        }
        private static int AlphebiticalDistance(string s, string t)
        {
            var result = 0;
            var i = 0;
            for(i = 0; i < Math.Min(s.Length, t.Length); i++)
            {
               // Console.Out.WriteLine($"loop 1 i={i} {s.Length} {t.Length}");
                result += Math.Abs(s[i] - t[i]);
            }
            for(; i < Math.Max(s.Length, t.Length); i++)
            {
               // Console.Out.WriteLine($"loop 2 i={i} {s.Length} {t.Length}");
                result += s.Length > t.Length ? s[i] : t[i];
            }
            return result;
        }
    }
}
/*
            var customers = new List<Customer>{
                new Customer{ Name = "Nathan", City = "New York"},
                new Customer{ Name = "Bob", City = "Boston"},
                new Customer{ Name = "Cindy", City = "Chicago"},
                new Customer{ Name = "Lisa", City = "Los Angeles"}
            };
*/
```