# Navico Pre-Interview Test

Please answer all questions to the best of your ability and provide as much detail as possible. We will discuss your answers in more detail if you are selected for a face to face interview.

**Candidate Name: Reza Amani**

## General

1. You have been asked to debug a random crash in a very large program with millions of lines of code. Initial analysis indicates that the crash occurs on a different line of code each time.

How would you debug this problem? Describe any tools that you may have used to aid with this type of investigation in the past

### Answer

|  |
| --- |
|  |

1. What is the most inventive or innovative thing you’ve done (in a SW development context)? **Pick one** and describe what problem were you trying to solve and the solution you came up with?

### Answer

|  |
| --- |
|  |

## C++ Programming

1. Which design patterns have you used in the past? Pick **one** and provide a **simple** C++ example of how this is used.

### Answer

|  |
| --- |
| Observer / publisher-Subscriber  In this design pattern, any publisher may report (i.e. publish) events and then those subscribers (observers) that have subscribed to published event will be notified and may execute a callback to react.  The main idea here is that publishers and subscribers don’t need to know each other (apart from a simple interface) and this helps reach better abstraction. It doesn’t matter how many observers have subscribed to a particular event. And even we can add or remove some observers in runtime. Because of independence between publishers and subscribers, each one can be reused independently.  Simple example in pseudo-code:  In the publisher side:  Class Publisher  {  List< Subscriber\*> subscriber\_list;  Public:  Virtual void Register(Subscriber\*); //adds the subscriber to the list of subscribers  Virtual void UnRegister(Subscriber\*); //removes the argument from the above list  Virtual void Update(); //calls the update function of all subscribers.  //If we have more than one event, the above functions should have another argument to determine the intended event  …  }  //Now any publisher inherits from the above class:  Class PublisherModule1 : public Publisher  {  ….  }  In the subscriber side:  Class Subscriber  {  Public:  Virtual void Update(Publisher\*); //reaction to the event. Having the publisher as an argument is needed only if more than one publisher can send the event to this subscriber (in order to distinguish between them)  …  }  Class SubscriberModule1: public Subscriber  {  …  }  In our project with C language, we implemented in another way; We had a 3rd module responsible for handling all stuff regarding events and subscriptions. It included an enum of all events and a list of callback functions from subscribers, Register functions that subscriber needed to link their callbacks to events, and finally ReportEvent functions that the publishers used to trigger the events. When an event was triggered, this coordinator module was responsible for calling the callback functions of associated subscribers. |
|  |

1. Imagine you are working in the team responsible for rendering charts on Navico’s products. A chart represents an area on a map (similar to Google maps). For the purpose of this exercise assume that a map consists of multiple charts drawn on top of each other, similar to the picture below.



Assume a chart has the following attributes.

* Position in space and size is specified by its top left coordinate (x1,y1) and its bottom right coordinate (x2,y2).
* Colour of the chart is specified in RGB, with each value between 0 – 255 (e.g. R=255, G=0, B=0 would imply a red chart)

1. Write a class called **Chart** that represents the above.
2. Write a class called **View** that can contain a maximum of 2 charts at a time
3. Implement a method **DoChartsOverlap()** in the **View** class which checks if the charts overlap.
4. Implement a method **GetColour(X,Y)** in the **View** class that will return the RGB colour of a given coordinate. If two charts overlap the colour of the point should be the average of the two colours.

You may assume the X and Y axis starts at 0 and has a maximum value of 100 for all of the above tasks.

### Answer

|  |
| --- |
|  |

1. How would you test your classes? Write some pseudocode for the unit tests you might write.

### Answer

|  |
| --- |
|  |