Analysis of Computer Programming & Development Degree Outcomes vs Hiring Requirements and Trends.

MAY 10, **2018**

The Forbidden Arts
Authored by: Jeremiah Stillings

The Forbidden Arts

A report of why graduates are being told NO.

This is the current degree's learning outcomes.

Nearly all technologies require software, and this degree provides students with the skills to develop that software for a broad range of devices. Students will learn how to write, test, and debug programs; how to use integrated development environments to create computer programs; and how to develop websites.

Upon completion of the Computer Programming and Development degree, a graduate should be able to:

- Use modern software development techniques and methodologies
- Use modern software testing techniques and methodologies
- Complete a software project from the definition phase through implementation

"I am running into programming tests as a gateway to being able to apply for a programming job, but I do not have the education to pass the tests. I quickly realized that I was unemployable in the current Computer Programming Industry" Jeremiah Stillings March 2018.

Contents

Course Learning Outcomes

Note: only courses related to this career field are listed.

FIRST SEMESTER

CTI 110	**CTI 110 – Web, Pgm, & Db Foundation**	**Credits: 3**
Understand basic concepts related in Website development Understand basic concepts and technologies related to computer programming Understand basic concepts and technologies related to databases		
CTS 115	**CTS 115 – Info Sys Business Concepts**	**Credits: 3**
Understand the relationship between business processes and IT Understand common IT concepts and technologies Understand business challenges and how information systems contribute to the decision-making process based on these challenges. Demonstrate knowledge of the potential offered by new technology and systems to business processes. Use technologies that improve workplace productivity.		
WEB 111	**WEB 111 – Intro to Web Graphics**	**Credits: 3**
Modify and optimize imagines optimized for web use. Demonstrate graphic design basics for the web, including color, contrast, readability, effective text, imagery. Utilize Photoshop and GIMP to process graphics for web use.		

SECOND SEMESTER

CSC 134	**CSC 134 – C++ Programming**	**Credits: 3**
Compile and Develop console programs. Understand data types, flow control streamed input/output and conversions. Declare, define and invoke functions. Use functions to manipulate and aggregate data. Understand string processing, exceptions handling, dealing with namespaces. Understand the object-oriented approach. Deal with classes and objects, class hierarchy and inheritance.		
CTS 120	**CTS 120 – Hardware/Software Support**	**Credits: 3** .
Describe functionality of PC hardware components. Differentiate between components, their purposes, and properties. Install and configure components. Use best practice maintenance procedures		
WEB 151	**WEB 151 – Mobile Application Dev I**	**Credits: 3**
Develop an app that makes use of the Android app activity lifecycle. Create an Android application that interacts with the user of the Android device. Create an app that uses Internet information sources and/or web services.		

THIRD SEMESTER

CSC 151	**CSC 151 – JAVA Programming**	**Credits: 3**
DBA 120	**DBA 120 – Database Programming I**	**Credits: 3**
Issue SQL commands that retrieve data based on criteria specified by the user. Use SQL commands to join tables and retrieve data from joined tables. Perform calculations based on data stored in the database. Use functions to manipulate and aggregate data. Use subqueries to retrieve data based on unknown conditions. Create, modify, and drop database tables. Manipulate data stored in database tables. Enforce business rules by using table constraints. Create users and assign the privileges users need to perform tasks.		
WEB 115	**WEB 115 – Web Markup and Scripting**	**Credits: 3**
Build a simple web site that organizes information effectively Use cascading style sheets to create style standards for a web site. Create a navigational framework that matches the content and genre of the site. Explain separation of concerns as it applies to the design and implementation of a web site Describe the issues involved in developing a web interface. Summarize the need and issues involved in web site implementation and integration. Explain why accessibility issues are an important consideration in web page development. Design and implement a web interface Explain and compare media file formats including lossy vs. lossless compression, color palettes, streaming formats, and CODECs		
WEB 182	**WEB 182 – PHP Programming**	**Credits: 3**

PHP installation/configuration process.

PHP data types and operators.

PHP functions and control structures.

PHP string manipulation.

Using PHP to manipulate files and directories.

Using PHP to manipulate arrays.

Using PHP in conjunction with MySQL.

Testing and debugging PHP

FOURTH SEMESTER

CSC 251	**CSC 251 – Advanced JAVA Programming**	**Credits: 3**
Write Java applications with a GUI. Write Java applications that interact with a Database. Write sound Java classes using advanced object-oriented techniques including inheritance and polymorphism.		
CSC 289	**CSC 289 – Programming Capstone Project**	**Credits: 3**
Apply fundamental business principles Design rough drafts of forms/screens/files Identify existing and potential problems Summarize solutions/findings Acquire feedback Apply logic Check accuracy Check user compatibility Consider hardware specifications Provide written documentation Explain documentation		
CTS 240	**CTS 240 – Project Management**	**Credits: 3**
Understand the genesis of project management and its importance Demonstrate knowledge of project management methods and techniques Understand and describe recent developments and research trends in the project management discipline. Use various project management applications to help plan and manage a technology project. Appreciate the importance of good project management		

Hiring Requirements

History of Hiring Requirements

Around 2007, companies learned that a lot of applicants could NOT program code at all and that they needed a tool to screen the massive amounts of applicants.

Meet FizzBuzz.

> Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

Figure 1 *https://imranontech.com/2007/1/24/using-fizzbuzz-to-find-developers-who-grok-coding/*

This new introduction of a test before you could interview was the beginning of many multi-million-dollar recruitment firms and the applicant’s worse fears.

The Current FizzBuzz

Currently tech companies use a paid service to vet applicants using online tests, if the programmer can pass the tests the service then allows the programmer to apply for a job with that tech company.

List of current paid services and companies they represent

https://triplebyte.com/

WE WORK WITH GREAT COMPANIES

We work with hundreds of companies of different sizes, stages, and industries and personally identify the ones that will be most exciting to you. All allow our pre-screened, pre-matched engineers to skip resume screens.

Dropbox Quip Quora reddit
instacart FLEXPORT OpenAI GUSTO asana
coinbase color coursera Opendoor LEVER
KHANACADEMY cruise twitch mixpanel Segment

https://www.interviewzen.com/

This is a customizable service that appears to be one of the original ones from 2007. This site does not appear to have a list of companies they represent.

https://www.codingame.com/work/solutions/coding-skill-assessment/

THEY TRUST OUR EXPERTISE

Adobe AMADEUS Booking.com ebay Nasdaq Nintendo WARNER BROS

https://codefights.com/

This was one of the first ones I ran into. They advertise on Facebook.

Some of our **Clients**

The world's best companies use CodeFights Recruiter to meet their technical hiring needs.

Dropbox Thumbtack EVERNOTE asana jet

ASCEND.IO Quora MuleSoft affinity WIZELINE

Dynamic Signal

https://mettl.com/en/coding-tests/

This looks like the largest client list yet.

Serving 1500+ Clients In Multiple Industries

Here's a closer look at how companies grow with Mettl's online assessments, platform and talent measurement.

OLX amdocs Microsoft FUJITSU

PIWIK AUTODESK BACKBASE enonic

accenture SYNTEL TOSHIBA wipro

https://www.hackerrank.com

These guys have an odd name, but lots of clients!

Thousands of customers trust HackerRank for tech recruiting.

Featured | Industries

Select Industry

High Tech

vmware — Screens 75% Faster

cisco — Attracted 1700 Security Engineers

<CODE_NATION> — Saves 300 Hours in University Recruiting

redhat — Reduced Interviews By Over 60%

Adobe

Microsoft

Symantec.

McAfee

intuit

fitbit

FUZEBOX

CITRIX

nitro

JUNIPER NETWORKS

EMC²

xerox

FireEye

riverbed Think fast.

Picking a side

The computer hardware industry has been a choice of PC or Mac since the beginning, but many people do not know that the software programming industry is in the same battle of having to pick a side.

Companies hire a Chief Technology Officer to make this call for their businesses.
Figure 2 *https://en.wikipedia.org/wiki/Chief_technology_officer*

The first choice is the Operating System the company will use.

Operating Systems

Windows
Mac
Linux
Mobile

- iOS
- Android

The second choice is based on the operating system choice. The CTO makes the call on the software stack and the company then hires programmers based on the software stack.

Microsoft's WINS Stack

WINS

Windows Server (operating system)
Internet Information Services (web server)
.NET (software framework)
SQL Server (database)

Microsoft's Programming Languages for WINS Stack

C#

Open Source's LAMP Stack

LAMP

- **Linux** (**operating system**)
- **Apache** (**web server**)
- **MySQL** or **MariaDB** (**database management systems**)
- **Perl**, **PHP**, or **Python** (**scripting languages**)

Open Source's Programming Languages for LAMP Stack

Perl, **PHP**, or **Python**

Java Platform

This is the odd ball! No technology stack needed.

Write once, run anywhere

Figure 3 *https://en.wikipedia.org/wiki/Write_once,_run_anywhere*

Java Platform Programming Languages

Java

As programmers, we will be required to program in many operating systems and would **benefit from an Operating Systems Class.**

As a college, you also will have to pick a technology stack to teach and pick a side, so to speak, as there is too much to teach if you dabble in all 3.

Java Platform IS easier for students to learn because the others undergo RAPID knowledge breaking changes. What worked last week may not work in the new patches. The Java platform currently releases on a 6-month update schedule.
For this reason, I recommend sticking with the Java Platform.

In the next section, I will go over what skills are needed to pass these online gatekeepers for each technology stack, in the order they were presented above.

Skills Needed to pass the tests

Microsoft's WIN Stack list of needed skills

General Software Development
Learn about general software development aspects, such as application lifecycle management and application specifications. These two concepts cover a general understanding of how software applications are created, deployed, and maintained by application developers.
Core Programming
Becoming successful in programming doesn't require a deep understanding of computer hardware, but a general understanding of how a computer stores and processes information helps the programmer become more efficient at writing code. In this module, viewers will be presented with core programming concepts, such as covering how computers store information using data, structures, and how algorithms are used to solve real-world problems in a computer application. Other core programming concepts are covered, in addition to showing how to use decision structures and repetition in your programs, plus a discussion on programming errors.
Object-Oriented Programming
Most programming tasks today revolve around object-oriented programming (OOP). This section covers how objects created in code can mimic real-world objects helping the programmer to better solve problems with their applications through modeling of the real-world objects represented in code.
Web Applications
This section will present the viewer with a general understanding of what web-based applications are and how they are used in providing functionality and services to users and other programs.
Desktop Applications
Although the Internet has created a move to more and more web-based applications and services, a fair bit of development is still accomplished for the desktop or laptop computer. This section presents on overview of the various types of applications a developer may be involved in creating for this platform.
Understand Databases
Almost all applications developed today rely on data of some kind. Programmers are expected to know what a database is and how to extract information from one. This section provides a high-level overview of databases and shows how to extract, insert, update, and delete data in database systems.

Figure 4 *https://mva.microsoft.com/en-US/training-courses/software-development-fundamentals-8248?l=D9b9nHKy_344984382*

Microsoft's certifications

Exam 98-361 would be my recommendation for completion of year 1

Figure 5 *https://www.microsoft.com/en-us/learning/exam-98-361.aspx*

Microsoft's list of deeper skills for a focus on Programming in HTML5 with JavaScript and CSS3

Module 1: Overview of HTML and CSS
This module provides an overview of HTML and CSS and describes how to use Visual Studio 2012 to build a Web application.
Lessons
- Overview of HTML
- Overview of CSS
- Creating a Web Application by Using Visual Studio 2012

Lab: Exploring the Contoso Conference Application

After completing this module, students will be able to:

- Describe basic HTML elements and attributes.
- Explain the structure of CSS.
- Describe the tools available in Visual Studio 2012 for building Web applications.

Module 2: Creating and Styling HTML5 Pages
This module describes the new features of HTML5 and explains how to create and style HTML5 pages.
Lessons
- Creating an HTML5 Page
- Styling an HTML5 Page

Lab: Creating and Styling HTML5 Pages

After completing this module, students will be able to:

- Create static pages using the new features available in HTML5.
- Use CSS3 to apply basic styling to the elements in an HTML5 page.

Module 3: Introduction to JavaScript
This module provides an introduction to the JavaScript language, and shows how to use JavaScript to add interactivity to HTML5 pages.
Lessons
- Overview of JavaScript Syntax
- Programming the HTML DOM with JavaScript
- Introduction to jQuery

Lab: Displaying Data and Handling Events by Using JavaScript

After completing this module, students will be able to:

- Explain the syntax of JavaScript and describe how to use JavaScript with HTML5.
- Write JavaScript code that manipulates the HTML DOM and handles events.
- Describe how to use jQuery to simplify code that uses many common JavaScript APIs.

Module 4: Creating Forms to Collect Data and Validate User Input

This module describes the new input types available with HTML5 and explains how to create forms to collect and validate user input by using the new HTML5 attributes and JavaScript code.

Lessons

- Overview of Forms and Input Types
- Validating User Input by Using HTML5 Attributes
- Validating User Input by Using JavaScript

Lab: Creating a Form and Validating User Input

After completing this module, students will be able to:

- Create forms that use the new HTML5 input types.
- Validate user input and provide feedback by using the new HTML5 attributes.
- Write JavaScript code to validate user input and provide feedback in cases where it is not suitable to use HTML5 attributes

Module 5: Communicating with a Remote Data Source

This module describes how to send and receive data to and from a remote data source by using an XMLHTTPRequest object and by performing jQuery AJAX operations.

Lessons

- Sending and Receiving Data by Using XMLHTTPRequest
- Sending and Receiving Data by Using jQuery AJAX operations

Lab: Communicating with a Remote Data Source

After completing this module, students will be able to:

- Serialize, deserialize, send, and receive data by using XMLHTTPRequest objects.
- Simplify code that serializes, deserializes, sends, and receives data by using the jQuery ajax method

Module 6: Styling HTML5 by Using CSS3

This module describes how to style HTML5 pages and elements by using the new features available in CSS3.

Lessons

- Styling Text
- Styling Block Elements
- CSS3 Selectors
- Enhancing Graphical Effects by Using CSS3

Lab: Styling Text and Block Elements using CSS3

After completing this module, students will be able to:

- Style text elements on an HTML5 page by using CSS3.
- Apply styling to block elements by using CSS3.
- Use CSS3 selectors to specify the elements to be styled in a Web application.
- Implement graphical effects and transformations by using the new CSS3 properties.

Module 7: Creating Objects and Methods by Using JavaScript

This module explains how to write well-structured and easily-maintainable JavaScript code, and how to apply object-oriented principles to JavaScript code in a Web application.

Lessons

- Writing Well-Structured JavaScript
- Creating Custom Objects
- Extending Objects

Lab: Refining Code for Maintainability and Extensibility

After completing this module, students will be able to:

- Describe the benefits of structuring JavaScript code carefully to aid maintainability and extensibility.
- Explain best practices for creating custom objects in JavaScript.
- Describe how to extend custom and native objects to add functionality.

Module 8: Creating Interactive Pages using HTML5 APIs

This module describes how to use some common HTML5 APIs to add interactive features to a Web application. This module also explains how to debug and profile a Web application.

Lessons

- Interacting with Files
- Incorporating Multimedia
- Reacting to Browser Location and Context
- Debugging and Profiling a Web Application

Lab: Creating Interactive Pages by Using HTML5 APIs

After completing this module, students will be able to:

- Use the Drag and Drop, and the File APIs to interact with files in a Web application.
- Incorporate audio and video into a Web application.
- Detect the location of the user running a Web application by using the Geolocation API.
- Explain how to debug and profile a Web application by using the Web Timing API and the Internet Explorer Developer Tools.

Module 9: Adding Offline Support to Web Applications

This module describes how to add offline support to a Web application, to enable the application to continue functioning in a user's browser even if the browser is disconnected from the network.

Lessons

- Reading and Writing Data Locally
- Adding Offline Support by Using the Application Cache

Lab: Adding Offline Support to a Web Application

After completing this module, students will be able to:

- Save and retrieve data locally on the user's computer by using the Local Storage API.
- Provide offline support for a Web application by using the Application Cache API.

Module 10: Implementing an Adaptive User Interface

This module describes how to create HTML5 pages that can dynamically detect and adapt to different devices and form factors.

Lessons

- Supporting Multiple Form Factors
- Creating an Adaptive User Interface

Lab: Implementing an Adaptive User Interface

After completing this module, students will be able to:

- Describe the need to detect device capabilities and react to different form factors in a Web application.
- Create a Web page that can dynamically adapt its layout to match different form factors.

Module 11: Creating Advanced Graphics

This module describes how to create advanced graphics for an HTML5 Web application by using a Canvas element, and by using Scalable Vector Graphics.

Lessons

- Creating Interactive Graphics by Using Scalable Vector Graphics
- Programmatically Drawing Graphics by Using a Canvas

Lab: Creating Advanced Graphics

After completing this module, students will be able to:

- Use Scalable Vector Graphics to add interactive graphics to an application.
- Draw complex graphics on an HTML5 Canvas element by using JavaScript code.

Module 12: Animating the User Interface

This module describes how to enhance the user experience in an HTML5 Web application by adding animations.

Lessons

- Applying CSS Transitions
- Transforming Elements
- Applying CSS Key-frame Animations

Lab: Animating User Interface Elements

After completing this module, students will be able to:

- Apply CSS transitions to elements on an HTML5 page and write JavaScript code to detect when a transition has occurred.
- Describe the different types of 2D and 3D transitions available with CSS3
- Implement complex animations by using CSS key-frames and JavaScript code.

Module 13: Implementing Real-Time Communications by Using Web Sockets

This module explains how to use Web Sockets to transmit and receive data between an HTML5 Web application and a server.

Lessons

- Introduction to Web Sockets
- Sending and Receiving Data by Using Web Sockets

Lab: Implementing Real-Time Communications by Using Web Sockets

After completing this module, students will be able to:

- Explain how Web Sockets work and describe how to send and receive data through a Web Socket.

- Use the Web Socket API with JavaScript to connect to a Web Socket server, send and receive data, and handle the different events that can occur when a message is sent or received.

Module 14: Creating a Web Worker Process
This module describes how to use Web Worker Processes to perform long-running operations asynchronously and improve the responsiveness of an HTML5 Web application.
Lessons
- Introduction to Web Workers
- Performing Asynchronous Processing by Using a Web Worker

Lab: Creating a Web Worker Process

After completing this module, students will be able to:

- Describe the purpose of a Web Worker process, and how it can be used to perform asynchronous processing as well as provide isolation for sensitive operations.
- Use the Web Worker APIs from JavaScript code to create, run, and monitor a Web Worker process.

Figure 6 *https://www.microsoft.com/en-us/learning/course.aspx?cid=2480*

Microsoft's certifications

Exam 70-480 would be my recommendation for this focus to be completed in year 2.

Figure 7 *https://www.microsoft.com/en-us/learning/exam-70-480.aspx*

Microsoft's list of deeper skills for a focus on Programming in C#

Module 1: Review of C# Syntax
This module reviews the core syntax and features of the C# programming language. It also provides an introduction to the Visual Studio 2012 debugger.
Lessons
- Overview of Writing Applications using C#
- Datatypes, Operators, and Expressions
- C# Programming Language Constructs

Lab: Developing the Class Enrolment Application

After completing this module, students will be able to:

- Describe the architecture of .NET Framework applications and use the features that Visual Studio 2012 and C# provide to support .NET Framework development.
- Use the basic data types, operators, and expressions provided by C#.
- Use standard C# programming constructs.

Module 2: Creating Methods, Handling Exceptions, and Monitoring Applications

This module explains how to create and call methods, catch and handle exceptions. This module also describes the monitoring requirements of large-scale applications.

Lessons

- Creating and Invoking Methods
- Creating Overloaded Methods and Using Optional and Output Parameters
- Handling Exceptions
- Monitoring Applications

Lab: Extending the Class Enrolment Application Functionality

After completing this module, students will be able to:

- Create and invoke methods, pass parameters to methods, and return values from methods.
- Create overloaded methods and use optional parameters and output parameters.
- Catch and handle exceptions and write information to the event log.
- Explain the requirement for implementing logging, tracing, and profiling when building large-scale applications.

Module 3: Developing the Code for a Graphical Application

This module describes how to implement the basic structure and essential elements of a typical desktop application, including using structures and enumerations, collections, and events.

Lessons

- Implementing Structs and Enums
- Organizing Data into Collections
- Handling Events

Lab: Writing the Code for the Grades Prototype Application

After completing this module, students will be able to:

- Define and use structures and enumerations.
- Create and use simple collections for storing data in-memory.
- Create, subscribe to, and raise events.

Module 4: Creating Classes and Implementing Type-safe Collections

This module explains how to create classes, define and implement interfaces, and create and use generic collections. This module also describes the differences between value types and reference types in C#.

Lessons

- Creating Classes
- Defining and Implementing Interfaces
- Implementing Type-safe Collections

Lab: Adding Data Validation and Type-safety to the Grades Application

After completing this module, students will be able to:

- Create and use custom classes.
- Define and implement custom interfaces.
- Use generics to implement type-safe collections.

Module 5: Creating a Class Hierarchy by Using Inheritance

This module explains how to use inheritance to create a class hierarchy and extend a .NET Framework class. This module also describes how to create generic classes and define extension methods.

Lessons

- Creating Class Hierarchies
- Extending .NET Framework Classes
- Creating Generic Types

Lab: Refactoring Common Functionality into the User Class

After completing this module, students will be able to:

- Define abstract classes and inherit from base classes to create a class hierarchy.
- Inherit from .NET Framework classes and use extension methods to add custom functionality to the inherited class.
- Create generic classes and methods.

Module 6: Reading and Writing Local Data

This module explains how to read and write data by using file input/output (I/O) and streams, and how to serialize and deserialize data in different formats.

Lessons

- Reading and Writing Files
- Serializing and Deserializing Data
- Performing I/O Using Streams

Lab: Generating the Grades Report

After completing this module, students will be able to:

- Read and write data to and from the file system by using file I/O.
- Convert data into a format that can be written to or read from a file or other data source.
- Use streams to send and receive data to or from a file or other data source.

Module 7: Accessing a Database

This module explains how to create and use an entity data model for accessing a database, and how to use LINQ to query and update data.

Lessons

- Creating and Using Entity Data Models
- Querying Data by Using LINQ
- Updating Data by Using LINQ

Lab: Retrieving and Modifying Grade Data

After completing this module, students will be able to:

- Create an entity data model, describe the key classes contained in the model, and customize the generated code.
- Use LINQ to query and work with data.
- Use LINQ to insert, update, and delete data.

Module 8: Accessing Remote Data

This module explains how to use the types in the System.Net namespace, and WCF Data Services, to query and modify remote data.

Lessons

- Accessing Data Across the Web
- Accessing Data in the Cloud

Lab: Retrieving and Modifying Grade Data in the Cloud

After completing this module, students will be able to:

- Use the classes in the System.Net namespace to send and receive data across the Web.
- Create and use a WCF Data Service to access data in the cloud.

Module 9: Designing the User Interface for a Graphical Application

This module explains how to build and style a graphical user interface by using XAML. This module also describes how to display data in a user interface by using data binding.

Lessons

- Using XAML to Design a User Interface
- Binding Controls to Data
- Styling a User Interface

Lab: Customizing Student Photographs and Styling the Application

After completing this module, students will be able to:

- Define XAML views and controls to design a simple graphical user interface.
- Use XAML data binding techniques to bind XAML elements to a data source and display data.
- Add styling and dynamic transformations to a XAML user interface.

Module 10: Improving Application Performance and Responsiveness

This module explains how to improve the throughput and response time of applications by using tasks and asynchronous operations.

Lessons

- Implementing Multitasking by using Tasks and Lambda Expressions
- Performing Operations Asynchronously
- Synchronizing Concurrent Access to Data

Lab: Improving the Responsiveness and Performance of the Application

After completing this module, students will be able to:

- Create tasks and lambda expressions to implement multitasking.
- Define and use asynchronous methods to improve application responsiveness.
- Coordinate concurrent access to data shared across multiple tasks by using synchronous primitives and concurrent collections.

Module 11: Integrating with Unmanaged Code

This module explains how to integrate unmanaged libraries and dynamic components into a C# application. This module also describes how to control the lifetime of unmanaged resources.

Lessons

- Creating and Using Dynamic Objects
- Managing the Lifetime of Objects and Controlling Unmanaged Resources

Lab: Upgrading the Grades Report

After completing this module, students will be able to:

- Integrate unmanaged code into a C# application by using the Dynamic Language Runtime.
- Control the lifetime of unmanaged resources and ensure that they are disposed properly.

Module 12: Creating Reusable Types and Assemblies
This module explains how to examine the metadata of types by using reflection, create and use custom attributes, generate managed code at runtime, and manage different versions of assemblies.
Lessons

- Examining Object Metadata
- Creating and Using Custom Attributes
- Generating Managed Code
- Versioning, Signing and Deploying Assemblies

Lab: Specifying the Data to Include in the Grades Report

After completing this module, students will be able to:

- Examine the metadata of objects at runtime by using reflection.
- Create and use custom attribute class.
- Generate managed code at runtime by using CodeDOM.
- Manage different versions of an assembly and deploy an assembly to the Global Assembly Cache.

Module 13: Encrypting and Decrypting Data
This module explains how to encrypt and decrypt data by using symmetric and asymmetric encryption.
Lessons

- Implementing Symmetric Encryption
- Implementing Asymmetric Encryption

Lab: Encrypting and Decrypting Grades Reports

After completing this module, students will be able to:

- Perform symmetric encryption by using the classes in the System.Security namespace.
- Perform asymmetric encryption by using the classes in the System.Security namespace.

Figure 8 *https://www.microsoft.com/en-us/learning/course.aspx?cid=2483*

Microsoft's certifications

Exam 70-483 would be my recommendation for this focus to be completed in year 2.

Figure 9 *https://www.microsoft.com/en-us/learning/exam-70-483.aspx*

Open Source's LAMP Stack list of needed skills

Introduction to Linux, Open Source Development, and GIT

Introduction
Open Source Software
Why Use Open Source Software?
How to Work in OSS Projects
Continuous Integration
OSS Licensing and Legal Issues
Linux and the Operating System
Graphical Environments and Interfaces
Getting Help
Text Editors
Shells, bash, and the Command Line
Filesystem Layout, Partitions, Paths and Links
System Components
System Administration
Essential Command Line Tools
Command and Tool Details
Users and Groups
Bash Scripting
Files and Filesystems
Linux Filesystems
Compiling, Linking and Libraries
Java Installation and Environment**
Building RPM and Debian Packages
Introduction to GIT
Git Installation
Git and Revision Control Systems
Using Git: an Example
Git Concepts and Architecture
Managing Files and the Index
Commits
Branches
Diffs
Merges
Managing Local and Remote Repositories
Using Patches
Closing and Evaluation Survey

Figure 10 *https://training.linuxfoundation.org/linux-courses/development-training/introduction-to-linux-open-source-development-and-git*

Apache

Apache History

Apache High Level Overview and Architecture

User and password management

Installing Apache

Apache Binaries

Directories and Permissions

Apache Modules
Filters
Handlers

Managing Apache

Starting Apache Manually and on System Start
Command Line Options
Server Status
Managing Apache Processes

Configuring Apache

Overview of httpd.conf
Document Root
Basic Server Directives
Logging Directives
Performance Tuning Directives

Security

Security Configuration Options
Passwords and Authentication
Allow and Deny Rules
Common Security Issues with Apache

Virtual Hosts

Virtual Hosting Overview
Name-Based and IP-Based Virtual Hosting

Content Caching

Caching Overview
Improving Caching Performance
Different Types of Caching

URL Mapping

Configuring Aliases and Redirects
The DocumentRoot Directive
Different Types of Caching

Figure 11 *http://www.learncomputer.com/training/apache/web-server-administration/*

MySQL

Explain MySQL storage engines
Explain database transactions
Obtain database metadata
Describe MySQL GUI tools
Monitor database performance
Perform database backup and recovery
Export and import database data
Describe the features and benefits of MySQL
Explain the basics of relational databases
Design an effective database
Build a database and tables by using SQL Modify or delete database entities
Query data with the SELECT command
Join data from multiple tables
Perform nested subqueries
Use built-in MySQL functions

Figure 12 *https://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=69&get_params=dc:D61918,clang:EN*

PHP

1 INTRO TO PROGRAMMING

A. Language and the Logic of Programming
B. Types of Programming Languages and Vocabulary
C. What is PHP
D. Why use PHP

2 PHP LANGUAGE BASICS

A. Syntax
B. Quotes
C. Comments
D. Special Characters
E. Data Types
F. Precedence
G. Symbols

3 PHP IDENTIFIERS

A. Variables
B. Constants
C. Arrays

4 PHP CONTROL FLOW BASICS

A. Operators
B. Conditionals

5 PHP FUNCTIONAL BASICS

A. Functions
B. Function Design Tools
C. File Systems (Handling)

6 FILE SYSTEM BASICS

A. Constants
B. Commonly Used File Functions
C. File System Performance

7 PHP WEB CONCEPTS

A. Client/Server Communications
B. How to embed PHP into HTML
C. How to embed HTML into PHP
D. Validating and Escaping
E. Cookies
F. Sessions
G. $_GET and $_POST

8 PHP DATABASE BASICS

A. Introduction to Relationship Databases
B. Keywords and Statements in SQL
C. Using PHP Functions with the MySQL Data Engine

9 DEVELOPING PHP APPS

A. General Software Development
B. Application Structure Guidelines
C. Organizing a concept solution
D. Application Skeleton
E. Debugging Techniques
F. Security

10 FINAL BINDINGS

A. OrderApp File Structure
B. OrderApp Architecture
C. OrderApp Request/Response Cycle
D. OrderApp Step Through
E. Resources
F. Course Summary

Figure 13 *http://www.zend.com/en/services/training/php-fundamentals-i*

Python 3

Control and Evaluations (25%)

- basic concepts: interpreting, compilation, language elements
- literals: Boolean, integer, floating point, string
- operators: priorities and binding
- numeric operators: **** * / % // + −**
- bitwise operators: **~ & ^ | << >>**
- string operators: *** +**
- Boolean operators: **not and or**
- relational operators (**== != > >= < <=**), building
- assignment and shortcut operators
- basic input and output: the **input()**, **print()**, **int()**, **float()**, and **str()** functions,
- conditional statements: **if**, **if-else**, and **if-elif-else**
- simple lists: constructing vectors, indexing and slicing, the **len()** function
- simple strings: constructing, assigning, indexing, slicing comparing, immutability
- building loops: **while**, **for**, **range()**, **in**, iterating through sequences
- controlling loop execution: **break**, **continue**

Data Aggregates (25%)

- strings in detail: ASCII, UNICODE, UTF-8, immutability, copying and cloning, advanced slicing, comparing string vs. string, string vs. non-string, basic string methods (**upper()**, **lower()**, **isxxx()**, **capitalize()**, **split()**, **join()**, etc) and functions (**len()**,**chr()**, **ord()**), escape characters
- lists in detail: indexing, slicing, basic methods(**append()**, **insert()**, **index()** etc) and functions (**len()**, **sorted()**, etc), the **del** instruction, iterating lists with the **for** loop, initializing, the **in** and **not in** operators, list comprehension, copying and cloning
- lists in lists: matrices and cubes
- tuples: indexing, slicing, building, immutability,

- directories: building, indexing, adding, and removing keys, iterating through keys and values, checking a key's existence

Functions and Modules (25%)

- defining and invoking your own functions and generators
- the **return** and **yield** keywords, returning results, the **None** keyword, recursion
- parameters vs. arguments, passing positional and keyword arguments, default parameter values
- name scopes, the **global** keyword
- defining and using **lambda** functions
- the **map()**, **filter()**, **reduce()** functions
- the **if** operator
- the import directives
- writing and using modules, the **__name__** variable
- creating and using .pyc files
- constructing and distributing packages, the role of the **__init__.py** file
- hiding modules' entities

Classes, Objects, and Exceptions (25%)

- defining your own classes
- class attributes: class variables and instance variables, defining, adding and removing attributes
- defining and using class methods, the meaning and usage of the *self* parameter
- inheritance and overriding, finding class/object components
- single inheritance vs. multiple inheritance
- name mangling
- the role of the **__str__** method
- introspection: **__dict__**, **__name__**, **__module__**, **__bases__** propertie s
- writing and using constructors
- the **hasattr()**, **type()**, **issubclass()**, **isinstance()**, **super()** functions
- using pre-defined exceptions, and defining your own exceptions

- the **try-except-else-finally** block, the **raise** statement
- the hierarchy of exceptions
- adding your own exceptions to an existing hierarchy
- assertions
- the anatomy of exception object
- input/output basics: opening files, stream objects, binary vs. text files, reading and writing files

Figure 14 *http://pythoninstitute.org/certification/*

Pearl

Useful for maintaining old systems, but realistically if they are running this, they already have someone to maintain it, and should not be looking to hire new talent to do so.

Java Platform list of needed skills

Oracle purchased Java from Sun Microsystems and Oracle SHOULD NOT be referenced as an authority of what is needed to know.

This is a link to entirely free course for Java 7 that covers 80% of what is needed to know for passing these gateway tests. Note Java 10 just released in March of 2018, but the concepts in the following table are still what one needs to know and are taught in this free book.

Online free book site
http://math.hws.edu/javanotes/

Linked PDF!
http://math.hws.edu/eck/cs124/downloads/javanotes7-linked.pdf

Producer/Consumer and Blocking Queues
Wait and Notify
Threads and Networking
The Blocking I/O Problem
An Asynchronous Network Chat Program
A Threaded Network Server
Using a Thread Pool
Distributed Computing
Network Programming Example
The Netgame Framework
A Simple Chat Room
A Networked TicTacToe Game
A Networked Poker Game

Advanced GUI Programming
Images and Resources
Images and BufferedImages
Working With Pixels
Resources
Cursors and Icons
Image File I/O
Fancier Graphics
Measuring Text
Transparency
Antialiasing
Strokes and Paints
Transforms and Shapes
Actions and Buttons
Action and AbstractAction
Icons on Buttons

Making Choices
Toolbars
Keyboard Accelerators
HTML on Buttons
Complex Components and MVC
Model-View-Controller
Lists and ListModels
Tables and TableModels
Documents and Editors
Custom Components
Finishing Touches
The Mandelbrot Set
Design of the Program
Internationalization
Events, Events, Events
Custom Dialogs
Preferences

This free course is extensive but **lacks** some theory that these gateway tests use.

Algorithms and Big-O notations

Regardless of what Software platform the college decides to teach, we need to know algorithms and Big-O Notation for the chosen language.

Here are the Java ones
As students we should be able to write a program using any of these sort algorithms and know the Big-O of each type.
https://github.com/TheAlgorithms/Java/tree/master/Sorts/src/sort

As students we should be able to write a program using any of these search algorithms and know the Big-O of each type.
https://github.com/TheAlgorithms/Java/tree/master/Searches/src/search

RegEx

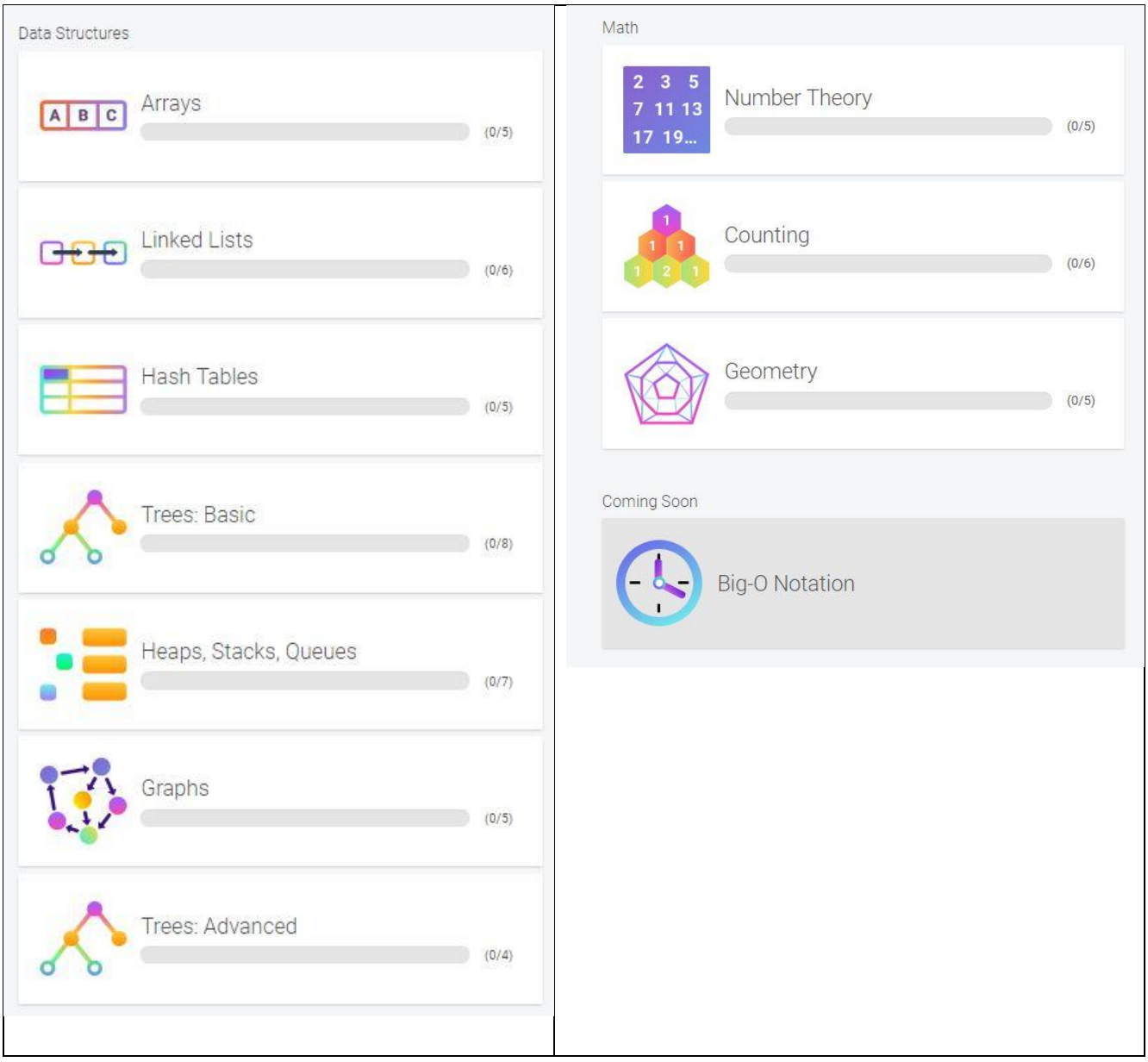
Regardless of what Software platform the college decides to teach, we need to know how to perform RegEx functions for the chosen language.

Here is a current Java tutorial on RegEx
https://www.javaworld.com/article/3188545/learn-java/java-101-regular-expressions-in-java-part-1.html

What do these gateway tests look like?

CodeFights.com

This is a picture of the categories that CodeFights tests you on



Sorting & Searching

Depth-First Search & Breadth-First Search (0/5)

Backtracking (0/5)

Sorting (0/7)

Dynamic Programming

Dynamic Programming: Basic (0/5)

Dynamic Programming: Advanced (0/5)

Special Topics

Common Techniques: Basic (0/7)

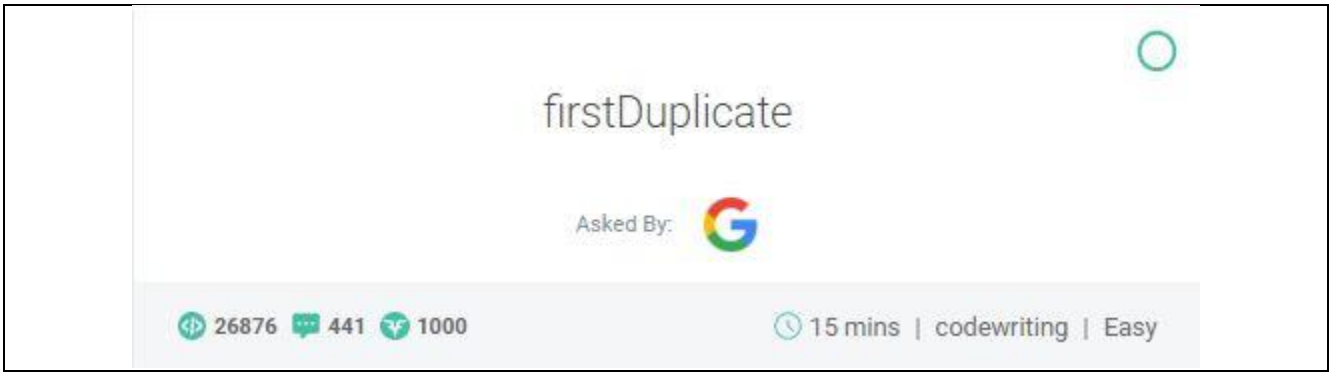
Strings (0/6)

Bits (0/5)

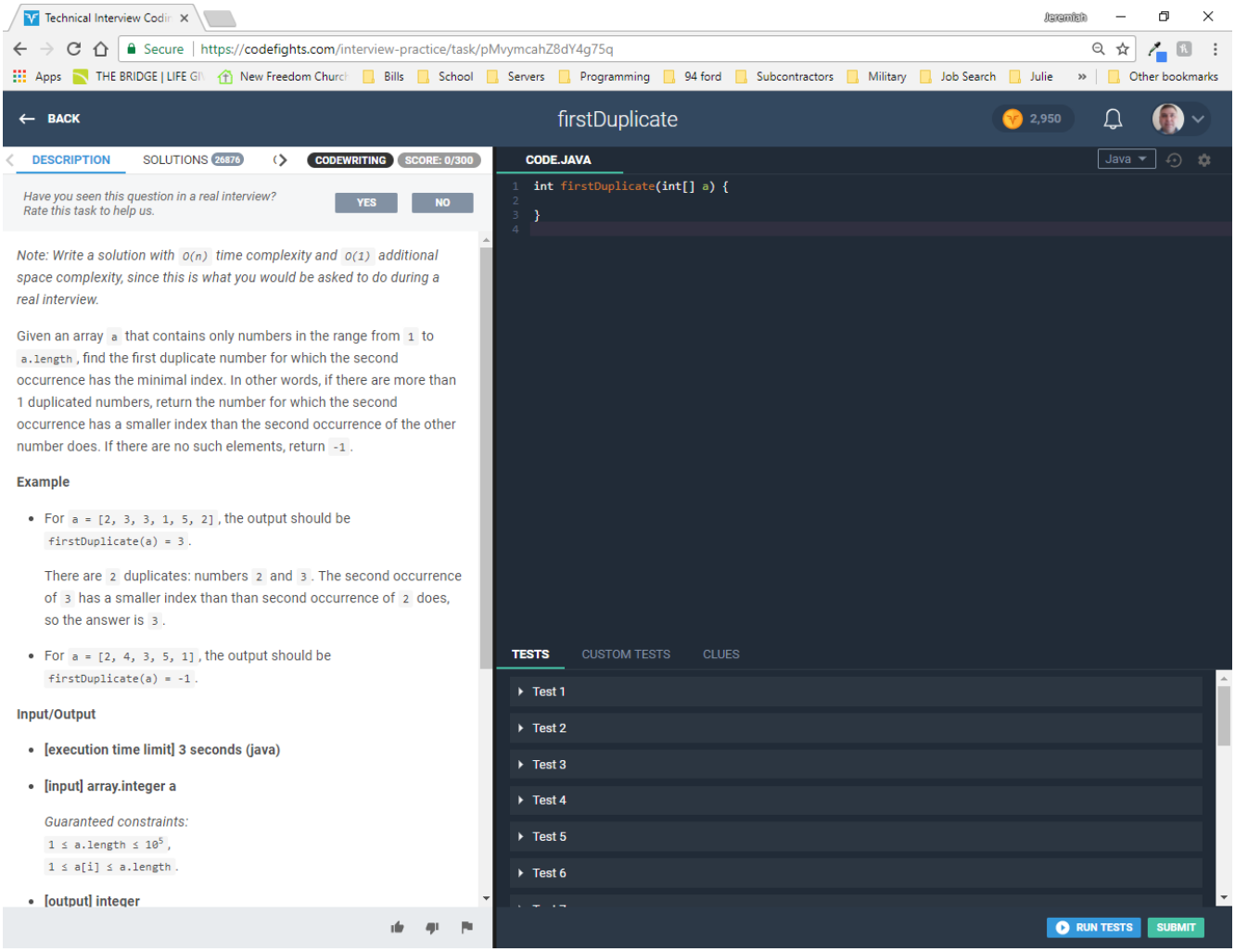
Common Techniques: Advanced (0/5)

RegEx (0/4)

The first one available to me is Arrays, the first task has a 15-minute timer and is marked EASY!



On the left is your instructions and on the right is your IDE, time is ticking as soon as it loads.



This is the instructions for task 1.

Note: Write a solution with `O(n)` time complexity and `O(1)` additional space complexity, since this is what you would be asked to do during a real interview.

Given an array `a` that contains only numbers in the range from `1` to `a.length`, find the first duplicate number for which the second occurrence has the minimal index. In other words, if there are more than 1 duplicated numbers, return the number for which the second occurrence has a smaller index than the second occurrence of the other number does. If there are no such elements, return `-1`.

Example

- For `a = [2, 3, 3, 1, 5, 2]`, the output should be `firstDuplicate(a) = 3`.
  There are `2` duplicates: numbers `2` and `3`. The second occurrence of `3` has a smaller index than than second occurrence of `2` does, so the answer is `3`.
- For `a = [2, 4, 3, 5, 1]`, the output should be `firstDuplicate(a) = -1`.

Input/Output

- **[execution time limit] 3 seconds (java)**
- **[input] array.integer a**

  Guaranteed constraints:
  $1 \le a.length \le 10^5$,
  $1 \le a[i] \le a.length$.
- **[output] integer**

  The element in `a` that occurs in the array more than once and has the minimal index for its second occurrence. If there are no such elements, return `-1`.

[Java] Syntax Tips

```
// Prints help message to the console
// Returns a string
//
// Globals declared here will cause a compilation error,
// declare variables inside the function instead!
String helloWorld(String name) {
    System.out.println("This prints to the console when you Run Tests");
    return "Hello, " + name;
}
```

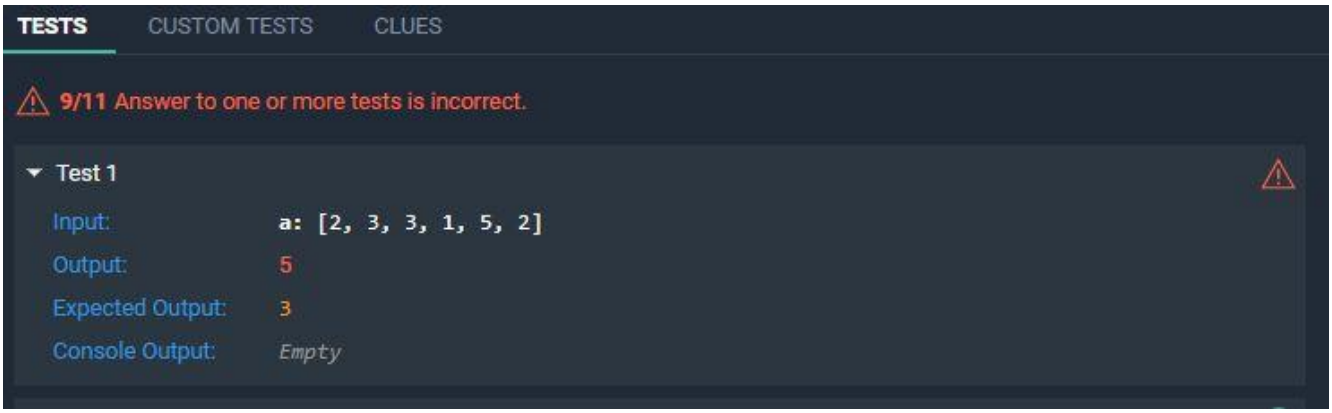
Our current education only covered 2 (green) of the 4 needed concepts.

Big-O Notation	Data Structure – Hash Map
For loop	If Statements

This was my solution that took me over **5 hours** of trial and error to come up with!

```
int firstDuplicate(int[] a) {
int answer = -1;
   int hit = 0;
   int holder = 0;
   int firstPlace = a.length;
   if( a.length > 1)
   {
      for(int i=0; i< a.length;i++)
      {
         for(int j = 1; j<a.length; j++)
         {
            if (a[i] == a[j])
            {
               if (a.length == 2)
               {
                  hit++;
                  if (hit == 2)
                  {
                     firstPlace = j;
                  }
               }
               else
               {
                  hit++;
                  if( hit >= a.length +1 )
                  {
                     if ( j <= firstPlace)
                     {

                        holder = j;
                        if(firstPlace > holder)
                        {
                           firstPlace = holder;
                        }
                     }
                  }
               }
            }
         }
      }
   }
   if( firstPlace == a.length)
   {
      answer = -1;
   }
   else
   {
      answer = a[firstPlace];
   }
   return answer;
}
```

My solution fails, and I have never been able to solve this in order to try more.



I found a solution here, but the idea of these sites is to see if YOU can program it, NOT GOOGLE, so it would be unethical for me to Google it and submit as my work.

https://mobile-security.ro/java-first-duplicate-index/

```
int firstDuplicate(int[] a) {

  Set<Integer> set = new HashSet<Integer>();

  HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();

  Map.Entry<Integer, Integer> min = null;


  for (int i = 0; i < a.length; i++){

    if(set.add(a[i]) == false && !hm.containsKey(a[i])) {

      hm.put(a[i], i);

    }

  }

  for (Map.Entry<Integer, Integer> entry : hm.entrySet()){

    if (min == null || entry.getValue() < min.getValue()){

      min = entry;

    }

  }

  return min == null ? new Integer(-1) : min.getKey();

}
```

TripleByte.com

Clicking sign up on another site, Triplebyte.com, I was presented with this.

Hey there!

I'm Mike, an engineer at Triplebyte.

Triplebyte's quiz is multiple-choice. It covers topics like basic programming, web development, basic algorithms and system design. Nobody's great at everything! The quiz finds your strengths.

Try a practice question:

What will the following code print?

```
var x = 1;
for (var i = 0; i < 3; i++) {
  x += 5 * i;
}
console.log(x);
```

- 1
- 10
- 16
- 31

You'll be able to choose front-end, mobile, or other special topics before we begin.

THAT WAS EASY :)

I did not even know what language this was. The function console.log() I was completely unfamiliar with. I had two thoughts, one, .log() accepted a parameter and returned a computation of a logarithm.

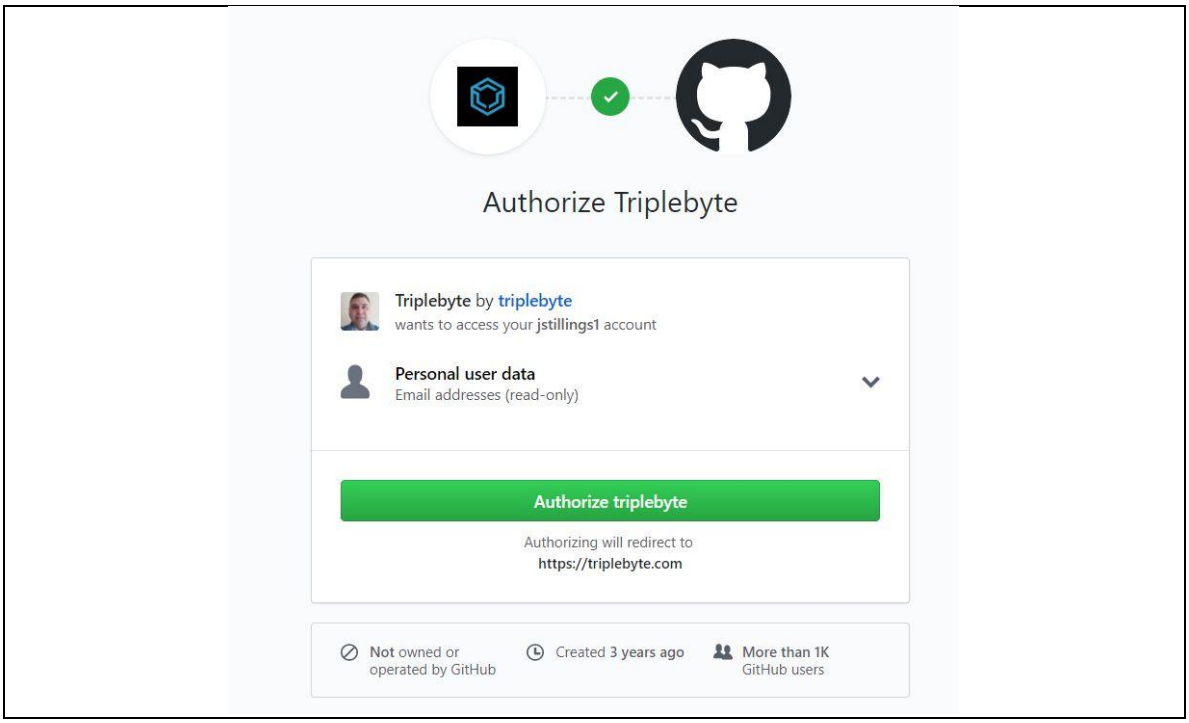
In mathematics, the logarithm is the inverse operation to exponentiation, just as division is the inverse of multiplication.

My other thought was that it was a print statement.
It turns out console.log() is debugging command in JavaScript.
https://www.w3schools.com/js/js_output.asp

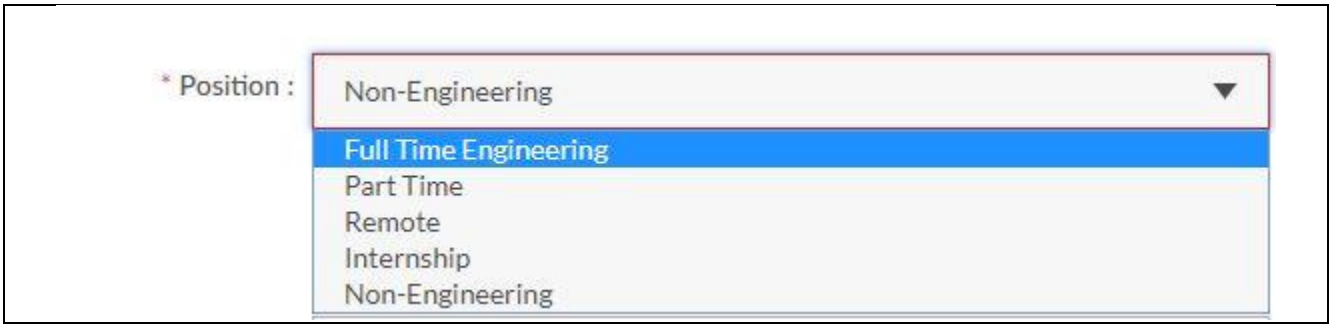
It took me about 6 minutes to come up with 16, which was the right answer and I progressed to the next screen.

Log in for a lot of these uses your Git Hub account, luckily, I have had one for years.

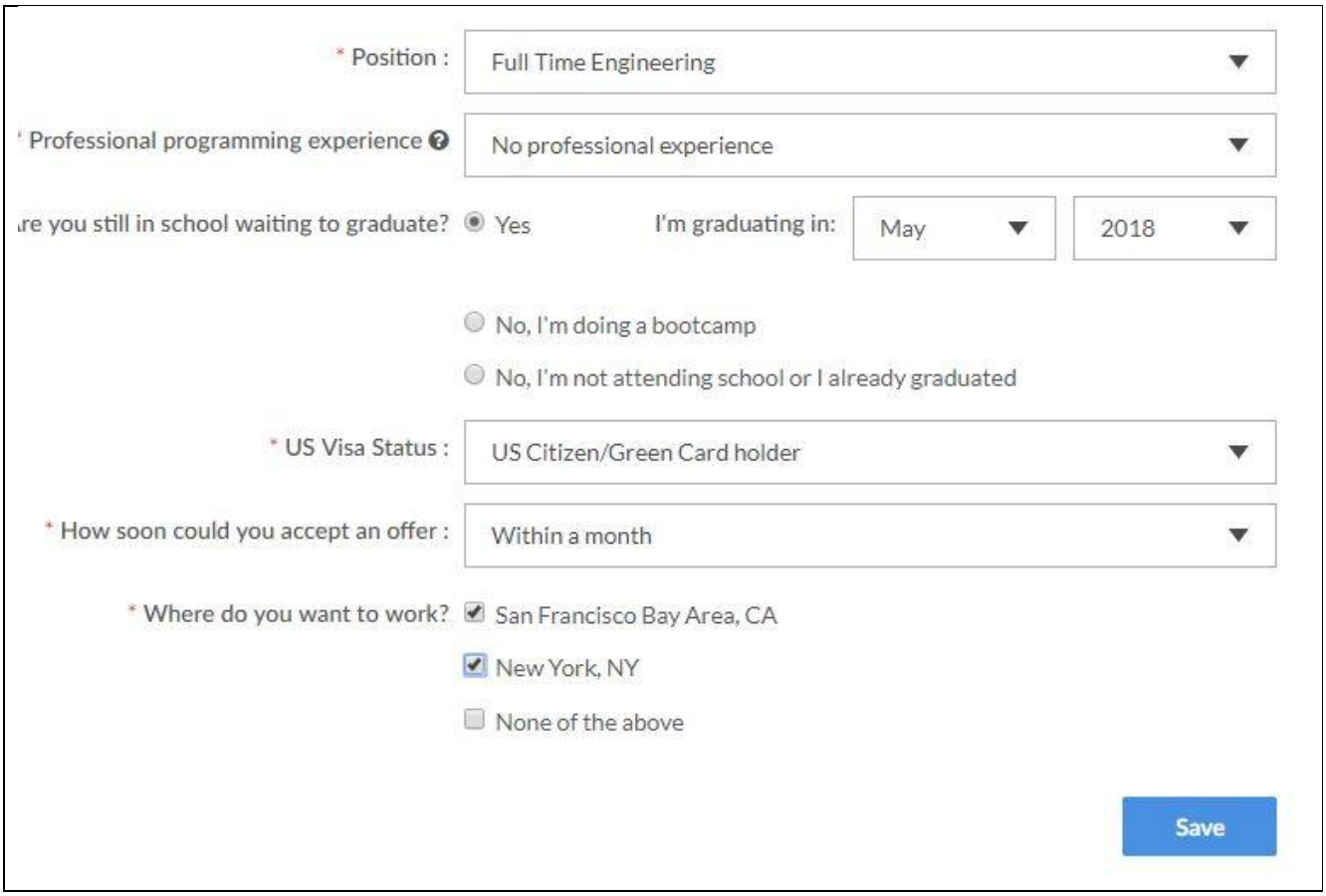
Git hub is not taught in the current curriculum but should be considered as adding.



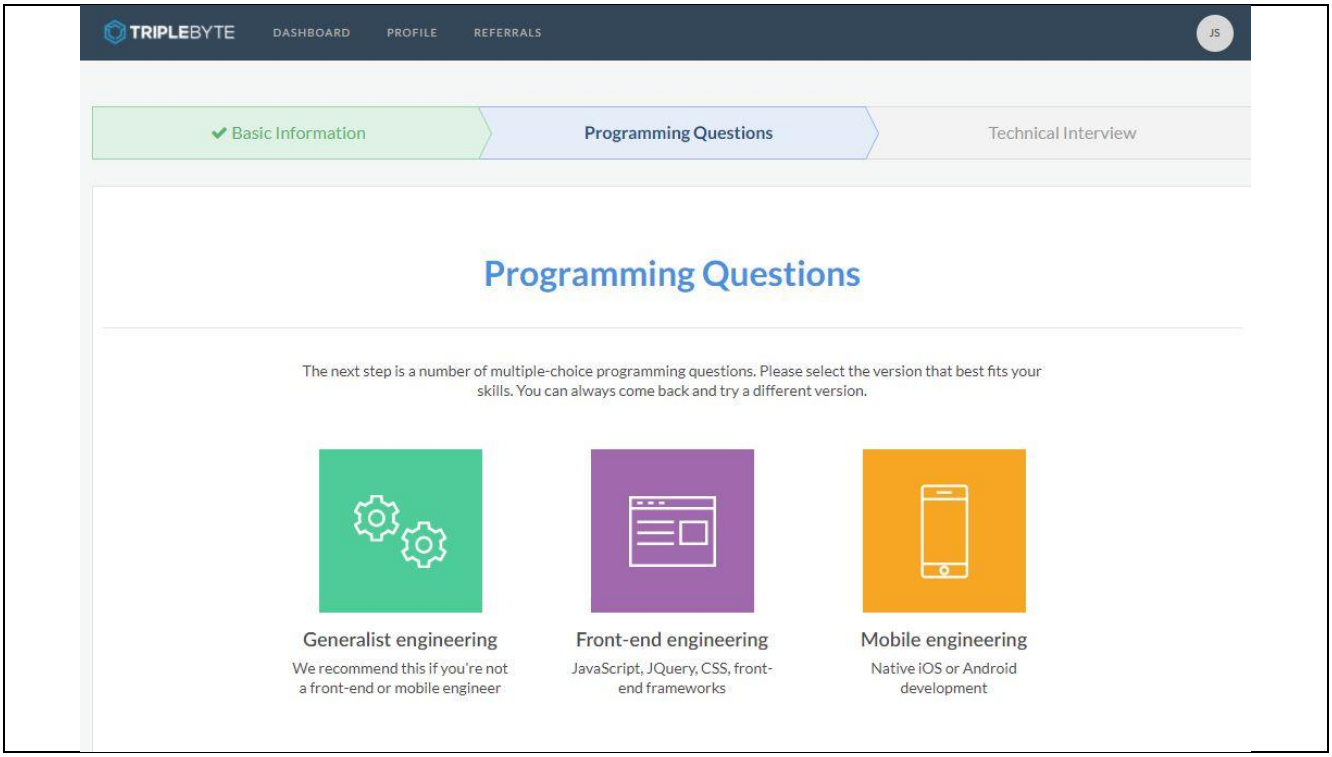
The next screen was a tad confusing. It wants me to list my position. But Programmer is not even selectable.



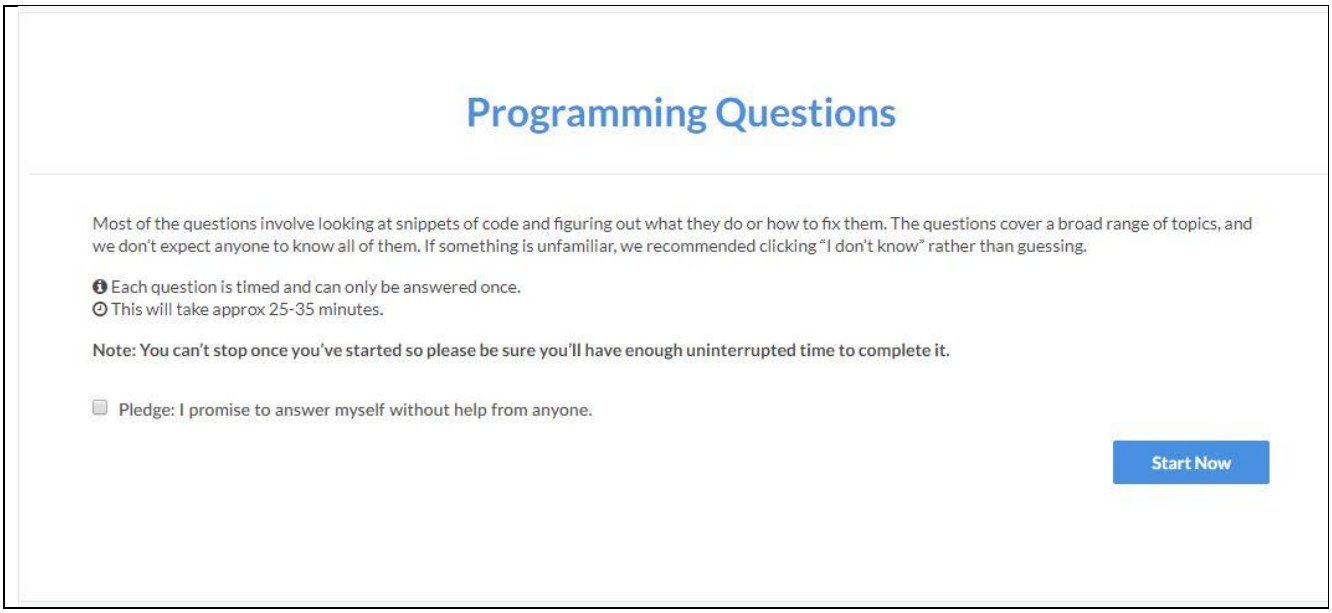
After a dozen more, pop ups saying they don't support that selection, I got this.



Once saved, I was presented with the programming questions.



I picked Generalist Engineering, which launched this screen.



Question 1 out of about 35

01 min 56 sec

The following code manages the position of the main character in a video game. Which of these statements about it is most accurate?

Language: Python

```python
def onFrame(self):
  self.character.velocity_y += 9.81
  self.character.y += character.velocity_y
  self.character.x += character.velocity_x

  # ...
```

Select the correct answer:

- If some frames take longer to compute than others, the character's vertical acceleration could be inconsistent, from one frame to the next.
- This code is memory inefficient. Rather than having separate variables — one for velocity_y, and one for velocity_x) — we should have a single variable, velocity, that tracks velocity across both axes.
- If some frames take longer to compute than others, the character's vertical acceleration could be inconsistent, from one point in time to the next.
- These operations are in the wrong order. We should first add the character's velocity to its position, and then add the acceleration constant to its velocity.

- I don't know

Question 2 out of about 35

01 min 55 sec

What's wrong with this code:

Language: Python

```python
# returns the content of all div tags on an HTML page as a list
def get_all_div_tag_contents(html_text):
    result = []
    html_tag_regex = "<([\w]+)([^>]*?)(([\s]*\/>)" + \
        "|(\>((([^\<]*?|<\!\-\-.*?\-\->)|(?R))*)\<\/\1[\s]*\>))"
    for tag_tuple in re.findall(html_tag_regex, html_text):
        result.append(tag_tuple[5])
    return result
```

Select the correct answer:

- It's impossible to parse an arbitrary HTML page with regular expressions. This code should be refactored to use a proper HTML parser.
- There's a missing closing bracket in the regex, HTML tags won't get matched with this regex.
- findall() returns a list, while finditer() returns a iterator. This could be a huge performance difference if you're going to handle big HTML pages. It's better to refactor this code to use finditer.
- tag_tuple[5] may throw an exception if tag_tuple has less than 6 elements, which may happen if div doesn't have any content.
- I don't know

Question 3 out of about 35

01 min 52 sec

Suppose you want to deploy a CPU-bound single-threaded app server to a machine with 16 logical CPU cores. Which design makes the most sense to maximize performance?

Select the correct answer:

- It's rarely helpful to run multiple copies of a CPU bound app server (the global interpreter lock means they just take turns). You'll want to make sure that TCP buffers are small, and that you're using fastCGI.
- You'll want a flexibly sized pool of server processes. This pool can grow if CPU utilization is low, and shrink if it's high. Then you can use round-robin DNS to balance load between them.
- You'll want your app server to `fork()` a new process on every request. This has better socket utilization and a lower memory footprint under load vs creating a fixed number of processes up-front.
- You'll want 16 copies of your app server, with a reverse proxy (like NGINX) in front to balance load and provide persistent connections and security.
- ✔ I don't know

Submit

Question 4 out of about 35

01 min 55 sec

Which of the following makes the most sense as part of scaling a SQL database to handle increased write load?

Select the correct answer:

- Writing to a materialized view, rather than to the main table
- Adding database replicas (in a master-slave configuration) to scale horizontally
- Adding database indices on the columns most often updated
- Removing little-used indices from the database and batching writes (where possible)

✔ I don't know

Question 5 out of about 35

01 min 46 sec

What is the value of g after the following code block runs?

Language: Javascript

```javascript
function f(x) {
  x *= 2;
  return function(y) {
    y *= x;
    return function(z) {
      return z * y;
    }
  }
}

let g = f(3)(4)(5);
```

Select the correct answer:

- 60
- 120
- An error occurs
- 5

✔ I don't know

Question 6 out of about 35

01 min 41 sec

What's wrong with this code?

Language: Java

```java
// return a string with the bits in binary
// of a 32-bit int
public static String getBitStringForInt(int n)  {
    StringBuilder rtn = new StringBuilder();
    for (int i = 31; i >= 0; i--) {
        int v = (1 << i) & n;
        rtn.append(v == 1 ? "1" : "0");
    }
    return rtn.toString();
}
```

Select the correct answer:

- The loop starts at 31, but should start at 32. Off by one
- The code checks the bits from right to left, but appends them from left to right
- The check v == 1 should be v != 0. The non-zero result will have 1 bit set, but in many different positions
- Everything's good
- I don't know

Question 7 out of about 35

02 min 52 sec

Suppose you're designing a distributed worker library, and would like it to be able to queue jobs using a number of different message queuing services (RabbitMQ, Amazon Simple Queue Service, ZeroMQ). What's a good way to handle making our code work with each of these services?

Select the correct answer:

- We can use a global "queuing_service" variable. This will be initialized to a flag like "rabbit_mq", or "amazon_sqs". Anywhere in our code where we need to interact with the queuing service, then, we can use a switch statement on this variable to make sure that we do the right thing.
- The best way is actually just to write 3 versions of the library (one for each of the queuing services). We'll end up with simpler (and faster) code in each case.
- As long as all our functions are referentially transparent, this is not really a problem. Referential transparency means that the order in which our methods are evaluated is irrelevant, and we can just run the code for all of our queuing services. The ones that are not set up will not cause any problems.
- We could design a base interface that defines how our library will interact with the queue service. We can create several implementations of this interface (one for RabbitMQ, one for ZeroMQ, etc). A method that runs when our library loads can look at config details, and instantiate the correct object.
- I don't know

Question 8 out of about 35

01 min 58 sec

What's the expected output of the following JavaScript code?

Language: Javascript

```javascript
function foo() {

  function bar() {
    setTimeout(
      () => console.log('Curly'),
      1000);
  }

  console.log('Larry');
  return bar;
}

let x = foo();
x();
console.log('Moe');
```

Select the correct answer:

- Moe, Larry, Curly.
- Curly, Larry, Moe.
- Larry, Moe, Curly.
- It won't compile.

- I don't know

This was the only question that I remotely thought about answering, but quickly confused myself with the lambada expression of the anonymous function, and the return of the bar with no ().

Question 9 out of about 35

02 min 55 sec

Imagine you're building a massively multi-player Pac Man game. You want hundreds of players to be able to play Pac Man against each other at the same time. You're going to pay prize money to the best players, so it's important that you limit their ability to cheat. How might you best build this?

Select the correct answer:

- Socket.IO is the way to go. Each client can open a web socket connection to the server, and transmit an event whenever their player moves. These events will be emitted to every other player in the game. That way every client will know where all the other players are, and can detect when they should die and remove themselves from the board
- None of these designs make sense
- A peer-to-peer architecture makes the most sense (the latency of a client-server version would be a problem). Each player could have a list of peers near them in the game, and send those peers their location every frame. You could have a voting system (like a blockchain) where clients vote to resolve state conflicts and agree on the final version.
- A server needs to run a canonical copy of the game, with all players. Each client can also run a copy of their region of the game, and use this to interpolate states between server updates. Clients would send their control inputs to the server (perhaps with a time stamp, although that raises cheating issues), and receive updates for all players near them.
- I don't know

Question 10 out of about 35

01 min 56 sec

In which of the following uses would a read–write lock most outperform a simple mutex?

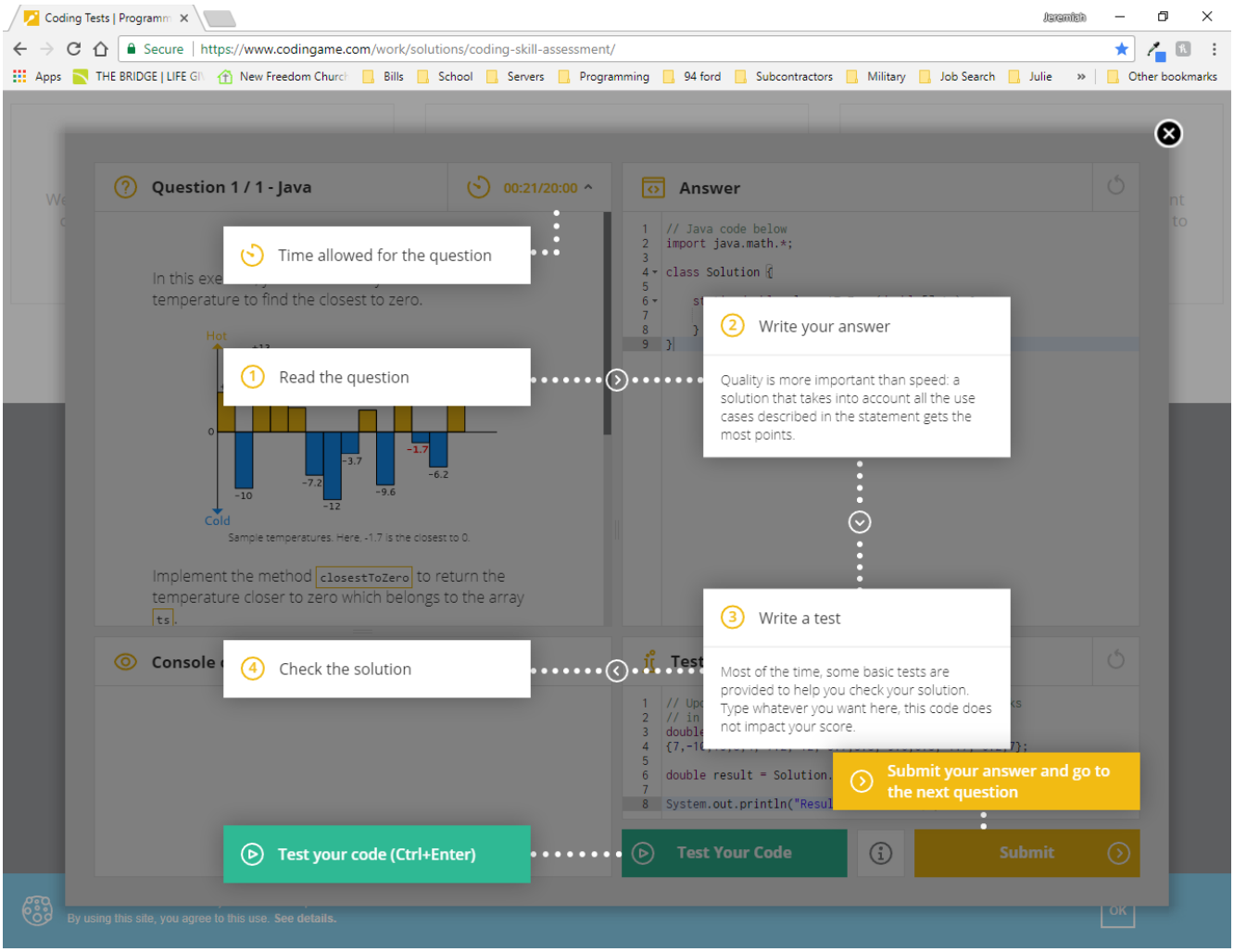
Select the correct answer:

- To control access to a dynamic list class in a read-heavy concurrent environment.
- To control access to a dynamic list class in a write-heavy concurrent environment.
- To control access to a critical section of very short duration (will be locked very briefly).
- To control access to a database from a cluster of worker processes.
- ✔ I don't know

After the 10 questions, I just hit log out, and never completed the sign-up process.

Codeingame.com

This was the DEMO for how their site works.



I gave it a shot.

Implement the method `closestToZero` to return the temperature closer to zero which belongs to the array `ts`.

- If `ts` is empty, return 0 (zero).
- If two numbers are as close to zero, consider the positive number as the closest to zero (eg. if `ts` contains -5 and 5, return 5).

Input:

- Temperatures are always expressed with floating point numbers ranging from -273 to 5526.
- `ts` is never `null`.

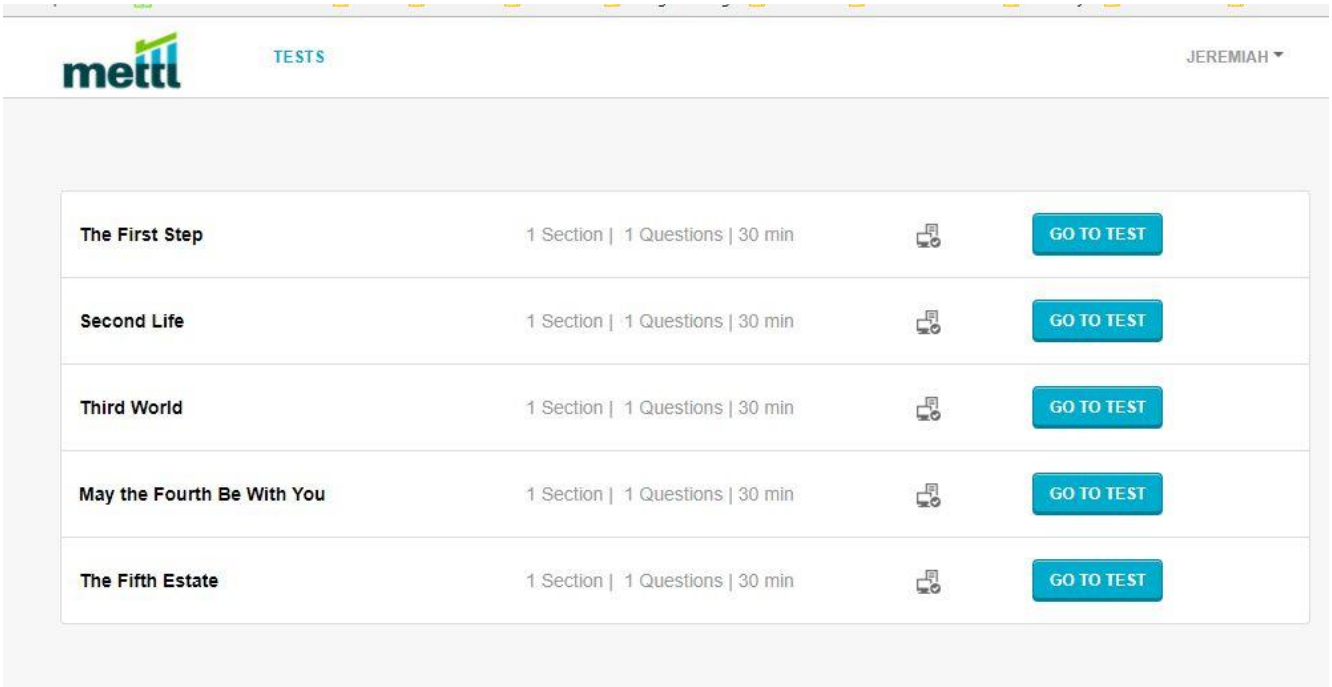
I spent about 5 minutes trying to solve this but had no idea how to go further.

Answer

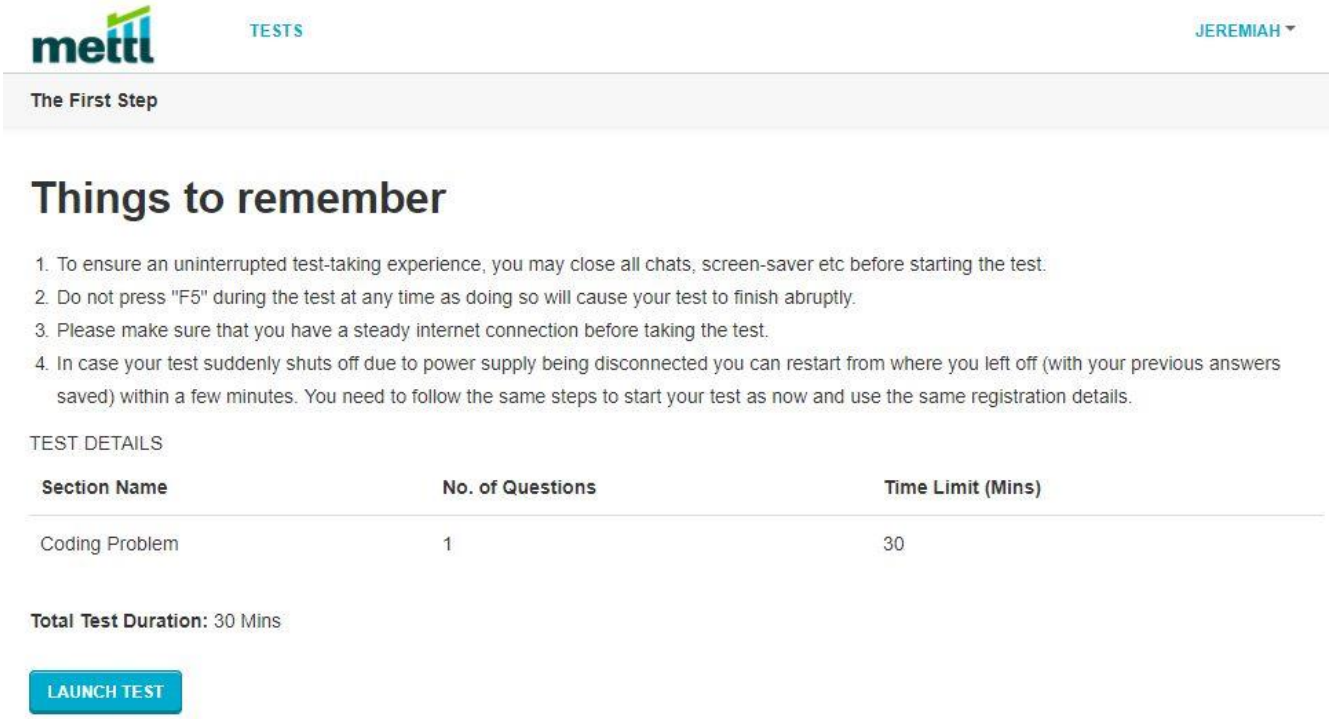
```java
// Java code below
import java.math.*;

class Solution {

    static double closestToZero(double[] ts)
    {
      double sendback = 0.0;
      if(ts.length() = 0)
      {
          sendback =0;
      }
      for( int i = 0; i < ts.length(); i++)
      {

      }
    }
}
```

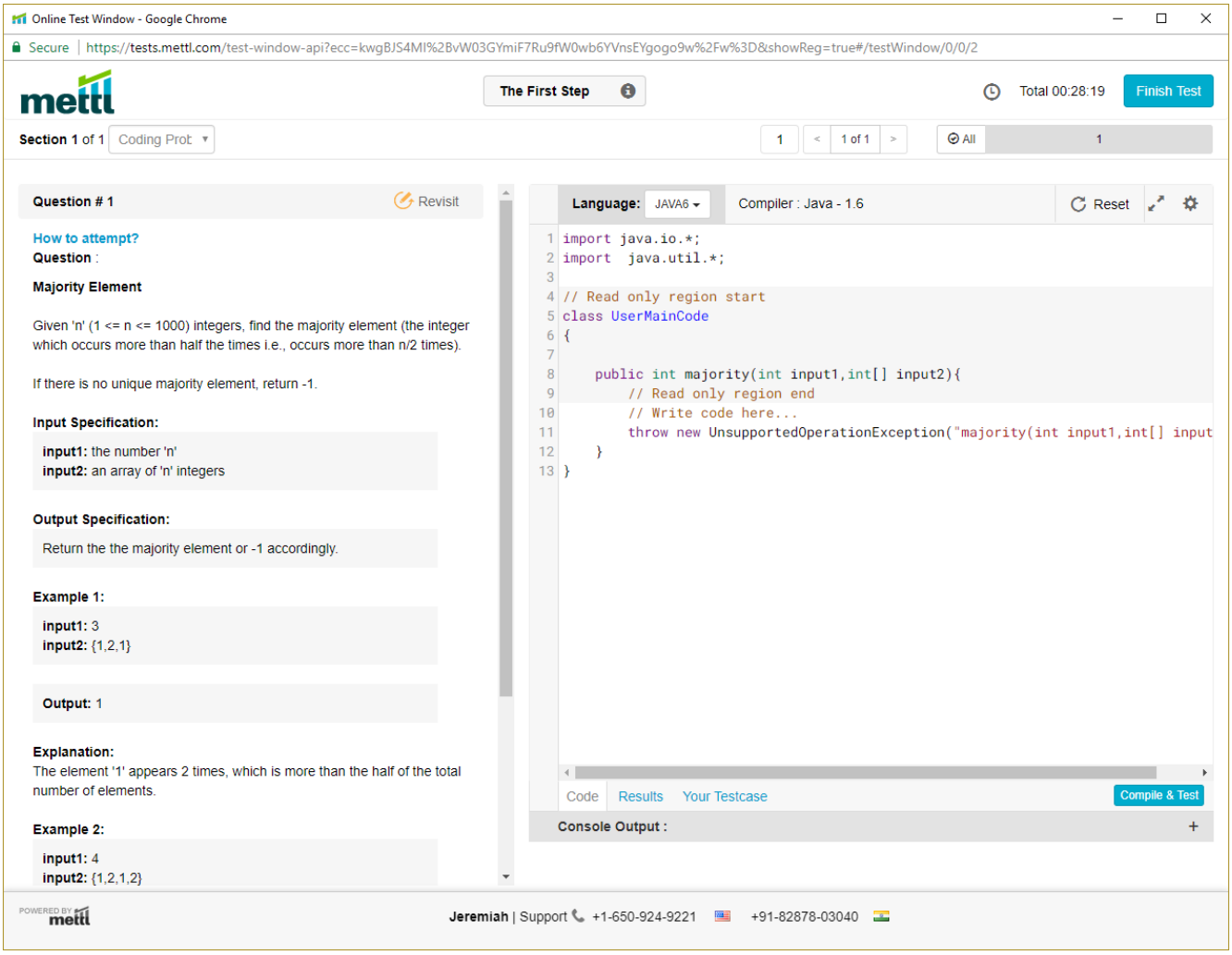
Mettle.com

This is the dashboard after signing up.



Clicking GO TO TEST displayed this screen.





I read the instructions on the left, and reread them, and then exited the test. I could not even comprehend what they were wanting me to do.

Conclusion

By now one should have come to understand that the Computer Programming & Development Degree learning outcomes of the courses contained herein are not providing the necessary concepts to be considered for employment as a computer programmer via the online vetting system that the hiring firms are using.