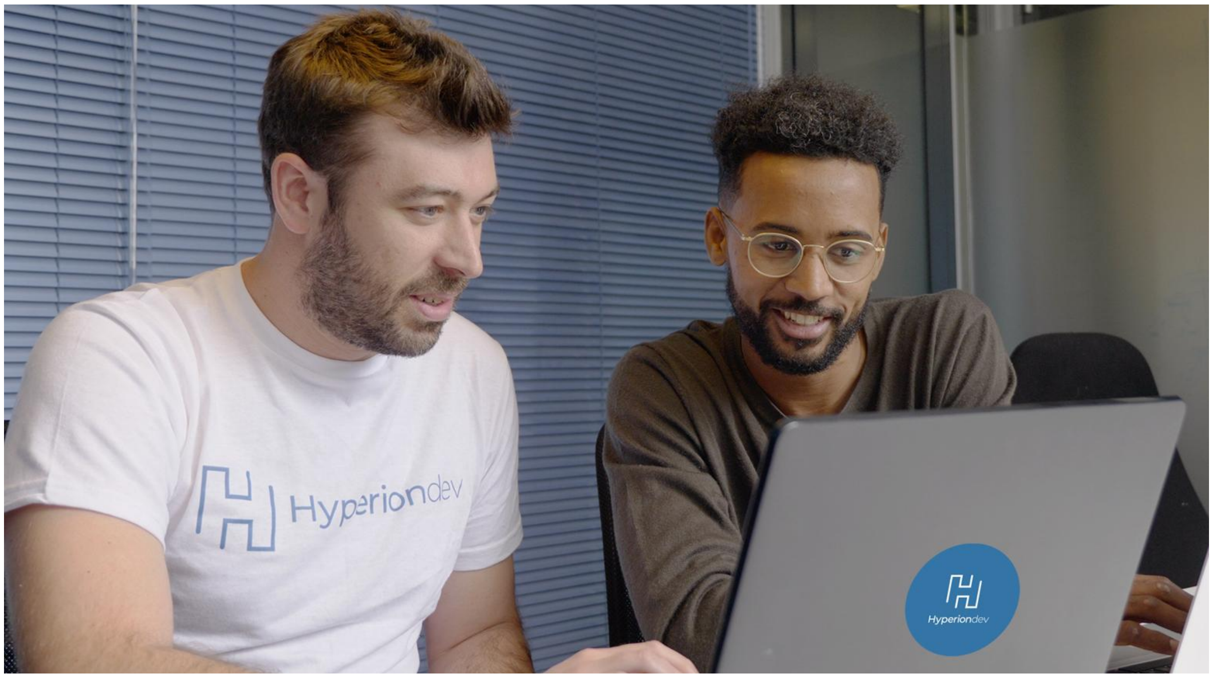
TAKE HOME TEST - WRITTEN ASSESSMENT

Code Reviewer

2021



Visit our website

Welcome

We're excited to get to know you and your skills better. The next step of the interview process with CoGrammar is to complete a take-home exercise. Please complete this exercise within **calendar days** of receiving it and make sure your responses are all sent through to the email ID from which you received this assessment unless otherwise specified.

Who are we?

CoGrammar is recognised as one of the top education technology startups in Europe, the Middle East, and Africa. Founded by Riaz Moola in 2012, CoGrammar's leadership team consists of ex-Google, Amazon, PwC, Yoco, Groupon employees, senior team members from GetSmarter (most valuable edtech startup in South Africa, acquired for >R1.5 billion) and graduates of the University of Cambridge, Cape Town, and Oxford. HyperionDev is our coding education product.

CoGrammar was funded by Facebook and Google in 2017 - winning first prize in Facebook's Africa Innovation Challenge Award as the top edtech startup on the African continent. CoGrammar works directly with Facebook's senior leadership - right up to Mark Zuckerberg himself - and through its headquarters in London is recognised as a leading edtech startup in Europe, the Middle East, And Africa. CoGrammar was recently recognised as one of the top 5 edtech startups in South Africa and is supported by top global edtech investors including the edtech fund behind Coursera, Andela, Udemy, & SoloLearn.

COGRAMMAR MEETS
MARK ZUCKERBERG
CoGrammar won first prize in the Internet.org 2016 Innovation Challenge edtech award and is financially backed by and partnered with Facebook.
facebook

We have built an online course platform that allows human code review to be scaled, applying this methodology to help thousands of students from over 30 countries learn how to code in a novel way. We pioneer effective and affordable software development education with this code review model, lowering the cost of access to tech careers around the world to shrink the tech skills gap and inequalities in the tech space.

Being a Code Reviewer

In this role, you will be joining our team of specialist Code Reviewers, also called CoGrammars. Our CoGrammars are elite, world-leading programming experts with a skill set that is at the intersection of technical coding skills. CoGrammars work with leading tech partners from around the world in fields as diverse as technical education, developer assessment, and tech team peer reviews.

Pursuing code review as a part-time opportunity provides a fulfilling way of specializing yourself further while garnering international work experience while being a full-time Code Reviewer is a promising career path with a trajectory similar to that of a traditional educator and compensation/benefits rivalling those in the software & IT industry.

Please complete the tasks below to help us understand how your skills may best fit the requirements of our team.

The Assessment

Please attempt the four Assessment Sections below, namely, marking tasks, full-stack, lightweight project and computational complexity.

To clarify, you will be submitting **five solutions in total**, one from Section A, the compulsory Section B, two from Section C, and the compulsory Section D. With the marking task, the key aspect is to guide the hypothetical student on the issue(s) they are facing without giving away the complete solution.

Please push your solutions to GitHub and kindly share the link(s) to the repositories. Alternatively, you may use Google Drive or Dropbox and send us the links in your email.

Assessment Sections

Assessment Section A: Marking Tasks (choose 1)

Task 1: Python Marking Task

Review and give feedback on a hypothetical student's code.

In this task, we will simulate a typical interaction that you might have with a student. You will be given a question that is asked by a hypothetical student as well as the student's submitted code. You will be required to answer the question and review the quality, structure and correctness of their code.

Let's say a student asks you the following question:

Hi There,

I'm having a bit of an issue with trying to figure out how to exactly implement the class methods so that I may call them outside of the class when trying to create a new SMS object.

Thanks!

The student's code can be found below. Answer the student's question and review the student's code, leaving specific comments that discuss the quality, structure and correctness of their code. Remember that the goal is to assist the student without giving them a complete solution.

```
# An SMS Simulation class SMSMessage(object):
hasBeenRead = False messageText = text fromNumber = number
def __init__(self,hasBeenRead,messageText,fromNumber):
    self.hasBeenRead = False self.messageText = text self.fromNumber = number

def MarkASRead(self):
    if userChoice == read:
        self.hasBeenRead = True

def add_sms():
def get_count():
def get_message():
def get_unread_messages():
def remove():

no_1 = SMSMessage(False, "Hello", "0798653452")
no_2 = SMSMessage(False, "WYD", "0845673864")
no_3 = SMSMessage(False, "How are you?", "0631873298")
```

```
SMSStore = [] userChoice = ""

while userChoice != "quit":
    userChoice = raw_input("What would you like to do - read/send/quit?")
    if userChoice == "read":
        # Place your logic here elif userChoice == "send": # Place your logic here elif userChoice
== "quit":
        print("Goodbye")
    else:
        print("Oops - incorrect input")
```

Task 2: Java Marking Task

Review and give feedback on a hypothetical student's code.

In this task, we will simulate a typical interaction that you might have with a student. You will be given a question that is asked by a hypothetical student as well as the student's submitted code. You will be required to answer the question and review the quality, structure and correctness of their code.

Let's say a student asks you the following question:

Hi there,

I am trying to do Task 6 and JOptionPane.ShowDialog() does not work. For some reason it does not store the input to the string variable it is assigned to, so my do-while loop ends up in an infinite loop. Can you please help me?

For context the task the student is working on is as follows:

- Create a new file called **do_whilePassword.java**.
- Imagine that the password to sign in to some computer is "John". Write a program that prompts the user to enter a password to login to this imaginary computer.
- If the user enters the correct password then notify the user and terminate the program.
- Should the user get the password incorrect three times in a row, then inform the user of the current password and allow the user to set a new password.

- Check the new password so that it satisfies these modified java identifier rules:
  - The password may have any number of characters between 1 and 20, inclusive.
  - The password may start with an underscore "_" or any letter of the alphabet.
  - The password may be any combination of upper and lower case letters.
  - The password consists only of numbers, English alphabet letters, and the underscore character.
- Should a proposed password not match these rules then inform the user that they have entered an invalid password and prompt the user to try again until an acceptable password is found.
- Here is an example run on the assumption that the current password is "John":

```
Prompt: Enter password:
      input: peter
Prompt: Incorrect password. Please enter password:
      input: luke
Prompt: Incorrect password. Please enter password:
      input: jane
Prompt: Password incorrect on three attempts.
Prompt: The password was John. Please set a new user password:
      input: john123
Prompt: password changed.
```

- Compile, save and run your file.

The student's code can be found below. Answer the student's question and review the student's code, leaving specific comments that discuss the quality, structure and correctness of their code. Remember that the goal is to assist the student without giving them a complete solution.

```
import java.util.*; import javax.swing.*;
public class do_whilePassword {
    public static void main ( String [] args ) {
        String choice = ""; do {
        //System.out.println( "Enter today's number from the menu:" ); choice =
JOptionPane.showInputDialog("Please enter in a password");
        }while(choice != "John" ); } }
```

Task 3: Web Dev Marking Task

Review and give feedback on a hypothetical student's code.

In this task, we will simulate a typical interaction that you might have with a student. You will be given a question that is asked by a hypothetical student as well as the student's submitted code. You will be required to answer the question and review the student's code.

Let's say a student asks you the following question:

Hi There, I currently have a moveable character on a webpage but I want to prevent it from going out of the screen's bounds. Please could you give me an idea and some basic code on how to approach this problem? Thanks!

The student's code can be found below. Answer the student's question and review the student's code, leaving specific comments that discuss the quality, structure and correctness of their code. Remember that the goal is to assist the student without giving them a complete solution.

Example.html

```html
<!DOCTYPE html>
<html>
    <head>
        <link rel='stylesheet' type='text/css' href='stylesheet.css'/>
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<!--Including jQuery -->
        <script type='text/javascript' src="script.js"></script> <!-- Including the
scripting file -->
    </head>
    <body>
        <img
src="http://staublicht.net/wordpress/wp-content/uploads/2011/08/walk_animation.gif"/>
    </body>
</html>
```

Script.js

```
$(document).ready(function () {

    alert("This page has loaded!");

    //Below is code which hides a paragraph when the button is clicked

$("button").click(function(){

    $("p").hide("slow", function () {

        alert("The paragraph is now hidden");

    });

}); //let width=Math.max($(document).width(), $(window).width());

//let height=Math.max($(document).height(), $(window).height()); let

height=$(window).height(); let width=$(window).width(); //Below is code which allows for

the character to move - why not try craft your own version?

$(document).keydown(function (key) {

//document.write(Math.max($(document).height(), $(window).height())); //

document.write(Math.max($(document).width(), $(window).width()));

switch (parseInt(key.which, 10)) {

    // Left arrow key pressed case 37:

    if ($('img').position().left > 0) {

        $('img').animate({

            left: "-=20px"

        }, 'fast');

    }

    break; // Up Arrow Pressed case 38:

    $('img').animate({

        top: '-=20px'

    }, 'fast');

    break; // Right Arrow Pressed case 39: if($('img').position().left<width){

    $('img').animate({

        left: '+=20px'

    }, 'fast');
}

break; // Down Arrow Pressed case 40:

$('img').animate({

    top: '+=20px'

}, 'fast');

break;

}

});

});
```

Stylesheet.css

```css
img {
    position: relative;
    left: 0;
    top: 0;
}

body {
    width: 100%;
    height: 100%;
    background: cyan;
    overflow: auto;
}
```

Task 4: Ruby Marking Task

Review and give feedback on a hypothetical student's code.

In this task, we will simulate a typical interaction that you might have with a student. You will be given code which a hypothetical student has written and be asked to review the code and to identify reasons it might not be working as the student anticipates.

The student has been given the task of creating a custom spell checker. This is an application that should take as input a string of words separated by spaces and any incorrectly spelt words should have an asterisk on either side in the output. Eventually, it should be able to check the words of common types of pets, namely cat, dog, rabbit, hamster, budgie and parrot, with the potential to expand this in the future.

The student has run into a problem which they don't understand. When given an input of "cat ct", the program should give "cat *ct*" as the output. However, it seems that the values aren't being checked, as they get "cat ct" as output.

The student's code can be found below. Identify the reason for the problem they are facing and suggest two ways in which they might overcome it, other than changing to a for loop. Then review the student's code, leaving specific comments that discuss the correctness of the remainder of the code, as well as its quality and algorithm.

```ruby
class Spelling_Checker

    def initialize()
      @words = ['catt', "ct", 'caaat','tcat']
    end

      def spellChecker(string)
        array = string.split(" ")

          array.each{ |n|
              if @words.include? n
                  "*#{n}*"
              end
            }.join(" ")
          end
        end


checker = Spelling_Checker.new
output = checker.spellChecker("cat ct")
p output
```

Assessment Section B: Full-Stack Task (compulsory)

Put your knowledge of web development to the test in this comprehensive technical task.

You can choose to design any web application that you like as long as it meets the following requirements:

- It is built using a backend framework in a programming language of your choice.
- It is built with any single-page application frontend framework in JavaScript, TypeScript or any other language of your choice.
- It creates, reads, updates and deletes information from a non-relational or relational database management system of your choice (choose 1).
- It authenticates users.
- The application allows for normal end-user access and admin access. An administrator should be able to monitor and make changes to users' behaviour.

Need some inspiration? Here are some examples of projects:

- An application that allows doctors to track information about patients and appointments. Here normal end-users may only be able to view appointments for a certain period of time (e.g. for a day or week) whereas an administrator can make, cancel or edit appointments and patient information.

- An application that a store owner can use to keep track of inventory. A normal end-user may be able to see when certain stock is running low and order new stock. An administrator would be able to add an item to the inventory, set the thresholds for stock (e.g. how much of a certain item to order at a time, when should new stock be ordered etc), decide to no longer stock certain items etc.

- An application that a conference centre could use to advertise upcoming events. Normal end-users might be able to see a list of upcoming events whereas an administrator might be able to add information about new events, cancel events, edit information about events etc.

- Your own implementation of an existing app. E.g. write your own web version of WhatsApp, Twitter or Instagram.

Submission Criteria

You will be required to code, test and debug an application of your choice.

You will need to submit:

1. The source code for your project. ***Set up a remote GitHub repository or two for your code and include access to these in your submission.*** Make sure that you have a corresponding excludes file (.gitignore) to point to your 'node-modules' directories. You will also need to submit compressed project folders for the front-end and back-end of your application with the 'node_modules' directories deleted. Ensure that your code meets the following criteria:
    a. The UI should be attractive, easy to use and intuitive..
    b. You should ensure that your application has been appropriately tested. You should include at least one snapshot test and appropriate unit tests for both the front-end and back-end of the application.
    c. The assessor should be able to launch your app by typing `npm start` from the command line interface.
    d. The file structure of the project should be well organised and easy to understand and use.
    e. Your code should be well documented with appropriate comments.
    f. The code should also be easy to read adhering to Google's style guide about indentation, meaningful variable and component names etc.
    g. Your code should be modular to make testing, debugging, code reuse and maintenance of your app easier.
2. A **readme.md** file that needs to include the following:
    a. An explanation of how to use the app.
    b. Clear instructions that an end-user will be able to follow to install, test and run your app on their local machine. This should include instructions for modifying any MongoDB URIs or API keys etc for your app.
    c. A description of the measures that you have taken to ensure the security of this app, including a description of how API keys have been dealt with.
    d. A description of any third-party APIs that you have used in your code.
    e. A description of where and how the application has been deployed. How has it been deployed? Back-end and front-end together or separate? Why?
    f. A link (or links) to the deployed app.

Assessment Section C: Programming Tasks

- 3 languages are provided in this section. Choose 2 out of the 3 languages and complete 1 task in each language.
    - For example: If you choose Python and Ruby for this section, then you should choose to complete 1 task in Python and 1 task in Ruby.
- Cannot be the same language you chose in Section A
- For the following tasks except for Python Task Option 2, please develop your solutions to the problems using TDD/BDD.
- Please use Git to keep track of the development history of your solutions.
- Please note that for this section, you will not only be assessed on the correctness of your solution but on your development process as well.

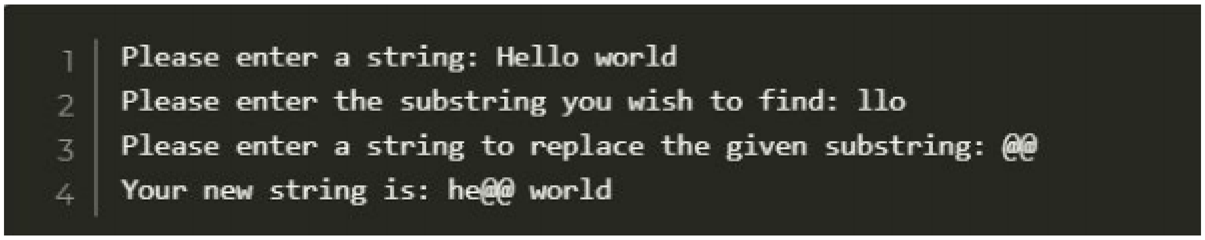
Python Task Option 1: Recursion

Put your knowledge of recursion to the test in this comprehensive technical task.

For this technical task, you are required to create a program that implements a search and replace function ***recursively***.

Your program should allow a user to enter a string, a substring they wish to find and another string they wish to replace the found substring with.

The output of your program should be similar to the output given below:



```
Please enter a string: Hello world
Please enter the substring you wish to find: llo
Please enter a string to replace the given substring: @@
Your new string is: he@@ world
```

Python Task Option 2: Book Review

Put your knowledge of data science to the test in this comprehensive technical task.

Below is some background information to help you better understand the concepts of Natural Language Processing and sentiment analysis.

For this task, you are required to use two different neural networks in Keras to try and classify a book review as either positive or negative, and report on which network type worked better.

For example, consider the review, "This book is not very good.'' This text ends with the words "very good" which indicates a very positive sentiment, but it is negated because it is preceded by the word "not", so the text should be classified as having a negative sentiment. We need to teach our neural network to recognise this distinction and be able to classify the review correctly.

This problem can be broken down into the following steps:

1. Get the dataset
2. Preprocessing the Data
3. Build the Model
4. Train the model
5. Test the Model
6. Predict Something

We will be working with real-world data in this task.

Get The Dataset

For this task, we will be using a small portion of the Multi-Domain Sentiment Dataset, which contains product reviews from Amazon. The full dataset contains reviews for products under the categories: kitchen, books, DVDs, and electronics, but we will only be looking at reviews for the book category.

We have two files, **positive.txt** and **negative.txt** (found here), containing the reviews. Each review is associated with a number of fields. The only field we are interested in is the “title” field, which contains the title of the review. We are going to use this title to predict the sentiment of the review. We could have used the review text itself, however, as this is a lot longer than the title, this is a much harder task.

For example, a review title might be “Horrible book”, whilst the review text might be “This book was horrible. If it was possible to rate it lower than one star I would have.”

Both the review title and the review text have the same sentiment — the title is just much more concise, which makes this task easier.

While some sentiments are easy to classify, like “don’t buy this horrible book”, others are less straightforward, like “'run don't walk to buy this book”. The latter is hard to classify because it contains the word “don’t”, which might be seen as an indication that this is a negative review, whereas actually it is a positive one (the reviewer is suggesting that you should go as fast as you can to get the book). Thus, sentiment analysis is not always straightforward — some samples will be easy to classify, while others will not.

Some code is already included in the notebook associated with this task to start you off, which loads the relevant part of the data (the review heading) and performs some preliminary preprocessing to remove strange characters.

It also is necessary to create a vocabulary (called a text corpus) — words which our neural network will know and to “tokenise” the input. If we have a review, such as “a good book”, it is necessary to turn this into a form that a computer can understand. First, each word in the dataset is mapped to a unique number in the vocabulary. A word tokeniser will then take a sentence like this and convert it to a sequence of numbers, which map to the relevant words in the vocabulary.

Eg: “a good book” becomes an array of numbers: [1, 12, 3]. This mapping means that “a” is the first word in the vocabulary, “good” is the twelfth and “book” is the third. This mapping depends on the dataset supplied to the tokeniser.

The code for tokenisation is already included, but it is important that you understand what it does.

Preprocessing the Data

In order to feed this data into our network, all input reviews must have the same length. Since the reviews differ heavily in terms of lengths, we either need to trim or pad the reviews so that they are the same length. For this task, we will set the length of reviews to the mean length, which is around 25 words. If reviews are shorter than 25 words we will need to pad them with zeros, if they are longer than 25 words we will trim them to this length by cutting off any words after this. Keras offers a set of preprocessing routines that can do this for us. In order to pad our reviews, we will need to use the `pad_sequences` function.

Build the Model

In the task today you will need to build a recurrent neural network to classify sentiment. The network will need to start with a special layer which will assist with text classification through a process called embedding.

Word embedding is a class of approaches for representing words and documents using dense vectors where a vector represents the projection of the word into a continuous vector space (Brownlee, 2017).

The position of a word within the vector space is learned from the text and is based on the words that surround the word when it is used. The position of a word in the learned vector space is referred to as its embedding.

Keras offers an embedding layer, used for neural networks on text data, and requires that the input data be integer encoded, so each word is represented by a unique integer (Brownlee, 2017). We have already achieved this format through tokenisation.

The embedding layer is trained as a part of the neural network and will learn to map words with similar semantic meanings to similar embedding-vectors. It is initialised with random weights and will learn an embedding for all of the words in the training dataset (Brownlee, 2017).

The Embedding layer is defined as the first hidden layer of a network. It must have three arguments (Keras Team, 2020):

- `input_dim:` This is the size of the vocabulary in the text data. For example, if your data is integer encoded to values between 0-5000, then the size of the vocabulary would be 5001 words.

- `output_dim`: This is the size of the vector space in which words will be embedded. It defines the size of the output vectors from this layer for each word. For example, it could be 32 or 100 or even larger. This is a hyper-parameter that needs to be tuned — test different values for your problem.

- `input_length`: This is the length of input sequences, as you would define for any input layer of a Keras model. For example, if all of your input documents are comprised of 25 words, this would be 25. This is the length which we padded/trimmed the inputs to during pre-processing.

Build a neural network, as outlined below.

- A recurrent neural network. This type of network is commonly used in NLP. This network should have the following architecture:

Embedding layer
SpatialDropout1D(0.2)
BatchNormalization()

LSTM(32)
Dense(2, activation='softmax')

Train and Tune the Model

You are now ready to train your model. Remember to compile your model by specifying the loss function and optimizer we want to use while training, as well as any evaluation metrics we'd like to measure — set the optimizer to 'adam' and the loss function to 'binary_crossentropy'.

Once compiled, you can start the training process. Note that there are two important training parameters that we have to specify, namely batch size and the number of training epochs. Together with our model architecture, these parameters determine the total training time.

For the network:

- Set the number of epochs to train for to 5 and batch size to 10.
- Tune the output_dim hyper-parameter of the embedding layer. Try values: 10, 25, 50 and 100. Report on the performance metrics for each value.
- Select the output_dim which gives the best performance on the test set and plot a graph of both the accuracy and loss of the model while training. Use these graphs to determine the point at which the model starts to overfit or if it has not yet converged. Identify a more optimal number of epochs to train for.
- You can also try tuning other metrics — such as batch size — to get the best possible performance.
- Report on the performance metrics of the final model.

Making Predictions

Finally, we would like to be able to use our model to predict something. To do this, we need to translate the sentence into the relevant integers and pad as necessary. This will allow us to put it into our model and see whether it predicts if we will like or dislike the book. A small selection of samples has been provided to get you going — you are welcome to add to this.

A notebook is associated with this task here, which contains some useful functions/code to get you started.

Follow these steps:

- Use the provided files **positive.txt** and **negative.txt** and follow the above steps to train a recurrent neural network. Your goal is to classify the sentiment of book reviews as positive or negative.
- Note: Some blocks of code are labelled do not modify — make sure that the code in these blocks is left alone, or else you will encounter issues.

- When complete, push this project to GitHub. Include a detailed README file in which you describe your project in detail.

Java Task Option 1: Multidimensional Arrays

Put your knowledge of arrays to the test in this comprehensive technical task.

For this task, you are required to create a program that allows you to play a text-based game of tic-tac-toe. Tic-tac-toe or noughts and crosses is a game where two players take turns marking an empty cell in a 3 x 3 grid with their respective tokens. Each player has either an X token or an O token. A player wins whenever they place three of their tokens in a horizontal, vertical, or diagonal row on the grid. A draw occurs when all the cells on the grid have been filled with tokens and neither player has won.

Your program should prompt each player to enter an X token or an O token alternately. The program should then redisplay the board on the console and determine the status of the game (win, draw or continue) whenever a token is entered.

The output of your program should be similar to the following:

```
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------

Player X, please enter a row (0, 1 or 2): 2
Player X, please enter a column (0, 1 or 2): 0

-------------
|   |   |   |
-------------
|   |   |   |
-------------
| X |   |   |
-------------

Player O, please enter a row (0, 1 or 2): 2
Player O, please enter a column (0, 1 or 2): 1

-------------
```

```
|   |   |   |
-------------
|   |   |   |
-------------
| X | O |   |
-------------

Player X, please enter a row (0, 1 or 2): 1
Player X, please enter a column (0, 1 or 2): 1

-------------
|   |   |   |
-------------
|   | X |   |
-------------
| X | O |   |
-------------

Player O, please enter a row (0, 1 or 2): 0
Player O, please enter a column (0, 1 or 2): 1

-------------
|   | O |   |
-------------
|   | X |   |
-------------
| X | O |   |
-------------
```

Java Task Option 2: OOP

Put your knowledge of object-oriented programming to the test in this comprehensive technical task.

For this task, you are required to create a program that simulates an ATM. Create a class called `Account` that contains:

- An int data field named `id` that stores the accounts unique identification number.
- A double data field named `balance` that stores the current balance of the account.
- A Date data field named `dateCreated` that stores the date on which the account was created.
- A constructor that creates an account with the specified id and initial balance.
- Methods that return the values of all data fields.
- Methods that set the values of the `id` and `balance` data fields.

- A method named `withdraw` that withdraws a specified amount from the `Account`.
- A method named `deposit` that deposits a specified amount into the account. Create two subclasses of the Account class; one for a checking account and one for a savings account. A checking account has an overdraft limit and a savings account can earn interest. Call these subclasses `SavingsAccount` and `CheckingAccount`.
- The `SavingsAccount` class should contain:
    - A double data field named `annualInterestRate` that stores the current annual interest rate.
    - A method named `getMonthlyInterestRate()` that returns the monthly interest rate. Monthly Interest Rate is calculated using `annualInterestRate / 12`. Note that `annualInterestRate` is a percentage so you need to divide it by 100.
    - A method named `getMonthlyInterest()` that returns the monthly interest. The monthly interest is calculated using `balance * monthlyInterestRate`.
- The `CheckingAccount` class should contain:
    - A double data field named `overdraft` that stores the overdraft limit.
    - A method that returns the value of the overdraft data field.
- Now use these classes to simulate an ATM:
    - Create five savings accounts and five checking accounts and store them in a list.
    - The system should prompt the user to enter an id. If the id is entered incorrectly, it should ask the user to enter a correct id.
    - Once an id is accepted, the main menu should be displayed as follows:
      ```
      Main menu
      1. check the balance
      2. withdraw
      3. deposit
      4. exit
      ```
    - The user should be able to enter 1 to view the current balance, 2 to withdraw money, 3 to deposit money, and 4 to exit the main menu. Once you exit, the system should prompt the user for an id again. Thus, once the system starts, it will not stop.

Ruby Task Option 1: Expanded form

For this task, you will be expected to write a program that will be able to write numbers in expanded form. In expanded form, a number is broken up according to its place value and each digit is expanded to show its value.

Eg. 5325 = 5000 + 300 + 20 + 5

As input, you will be given an integer. The expected output should return a string.
For example:

```
expanded_form(12);    // Should return '10 + 2'
expanded_form(42);    // Should return '40 + 2'
expanded_form(80504); // Should return '80000 + 500 + 4'
```

You may assume that all numbers will be whole numbers greater than 0.

Please name your method as demonstrated above. Your solution will be run against a series of tests.

Ruby Task Option 2: Case conversion

For this task, you will be expected to swap the case of a letter depending on the binary representation of the supplied number.

Eg: Given ("mine", 11) output: "MiNE"

11 can be represented as 1011 in binary. Thus, all letters in positions corresponding to ones need to have their case swapped.

Further requirements:

- When the word is longer than the binary representation, the binary representation should be repeated.
  Eg: Given ("absoluTE", 9) output: "AbsOLuTe"
- All characters that do not belong to the alphabet (the set a-z) should be skipped.Eg: Eg: Given ("absoluTE absoluTE", 9) output: "AbsOLuTe AbsOLuTe"

As input, you will be given a word and an integer. The expected output should return a string with the case toggled according to the binary representation of the supplied number.
For example:

```
swap("absoluTE", 4)  // Should return "AbsOlutE"
swap("Mine", 11)     // Should return "miNE"
swap("no ways", 11)  // Should return "No WAYs"
```

Please name your method as demonstrated above. Your solution will be run against a series of tests.

Assessment Section D: Computational Complexity (compulsory)

Put your knowledge of computational complexity to the test in this comprehensive technical task.

For this section, you are required to report on the space and time complexity of `doBracketsMatch`, the function defined in the JavaScript code below. Your report should include clear justifications for your solution.

```javascript
function BracketStack() {
 let openBracketsCount = 0

 this.isEmpty = function() {
   return openBracketsCount === 0
 }

 this.push = function() {
   openBracketsCount++
 }

 /**
  * @throws If stack is empty
  */
 this.pop = function() {
   if (this.isEmpty()) throw new Error("Cannot pop empty stack")
   openBracketsCount--
 }
}

/**
* @param {string} symbol
* @returns A function that checks if a given value is the symbol
*/
function isSymbol(symbol) {
 /**
  * @param {string} value
  * @returns {boolean}
  */
 function check(value) {
   return symbol === value
 }
 return check
}
```

```
/**
* Checks if a pair of brackets match
* @param {string} inputString
* @param {string} openingSymbol
* @param {string} closingSymbol
*/
function doBracketsMatch(inputString, openingSymbol, closingSymbol) {
 let stack = new BracketStack()
 let isOpeningSymbol = isSymbol(openingSymbol)
 let isClosingSymbol = isSymbol(closingSymbol)
 for (let i = 0; i < inputString.length; i++) {
   let value = inputString[i]
   if (isOpeningSymbol(value)) stack.push()
   if (isClosingSymbol(value))
     if (stack.isEmpty()) return false
     else stack.pop()
 }
 return stack.isEmpty()
}
```

Task Submission

Please push your solutions to GitHub and kindly share the link(s) to the repositories. Alternatively, you may use Google Drive or Dropbox and send us the links in your email.

You would also be able to respond in email with all questions you have around the assessment. The hiring manager will be able to then provide you with some guidance, resources and answers (if possible) around your query. The aim is for you to succeed!

This assessment is designed to understand you better as a candidate and has no right or wrong answers. We will be weighting the assessment based on the following:

1. The thoroughness of your submission
2. Use of the data provided
3. Creativity
4. Research efforts
5. Presentation
6. Completed within the given time

Thank you for your interest in joining our team at CoGrammar!