Assessment	Weight Towards Course Grade	Due Date
Assignment 1	15%	See Moodle
Group Size	**Submission Type**	**Topics Covered**
Individual	PDF	Formal Testing Techniques API Testing Integration Testing

Submission Instructions

1. Put all answers in a PDF file
2. Include your student name and id in the file
2. Upload PDF file to Moodle.

Before uploading you file:

1. Ensure that your report is **neat, easy to read, and properly formatted.**
2. Check your spelling and grammar before submission.

Assignment Description

In this assignment, you will explore how to design and execute tests against a given software system.

This assignment consists of

Problem Description

Information about the Pokemon game is given on page 7.

Use the following information to answer Questions 1-3

The current game contains 150 Pokemon. Each Pokemon is given a 4 digit id number, starting with 0001. The ids increment by 1.

In January 2020, Nintendo plans on releasing a new version of the Pokemon video game. The new version has a feature that lets players add custom Pokemon to the game. Once a custom Pokemon is added, the Pokemon is available to all players. The custom Pokemon cannot be updated or removed from the game.

You work as a Test Engineer on Nintendo's API team. Your job is to test Nintedo's Pokemon APIs.

1.   What endpoints should you test to ensure that players can use the game's new feature? Write your answer in the space below:

Endpoint	Request Type

2. What are the 3 things must you test to ensure that you have properly tested the endpoints? Remember - you are testing endpoints that allow the player to use the new game feature!

3. Suppose you are using Postman to test the endpoint for adding a new custom Pokemon to the database. Write the Postman test case code for the endpoint.

In your test case, you should include all appropriate assertions needed to properly test the endpoint.

As a reminder, the built-in Postman functions include:

- tests

- responseCode[“code”]
- responseBody
- JSON.parse()

4. As discussed in class, an important part of API testing is to ensure that a command actually gets executed on the backend database. What strategy can you follow to creat a test case that checks if a database operation actually occurs?

==

Use the following information to answer Questions 5-6

Nintendo wants you to test the **<u>system</u> that calculates the number of experience points** that are required for a Pokemon to reach the next level. A diagram of the overall system architecture is shown in ***Technical Information - Architecture Diagram*.**

To make your job easier, Nintendo only wants you to check if the function works for Pokemon moving from level 2 to 3.

5. Examine the Architecture diagram. What test cases should you create to ensure proper functional coverage of the system?

Test Case	Description of test case
1	
2	
3	
4	
5	

Assume you are using equivalence testing (black box) to choose inputs for testing the system.

6. What equivalence partitions do you need to create?

7. What inputs do you need to select to test using partitions. Fill in the details of your inputs in the chart below:

Equivalence Class	Input	Valid/Invalid Input	Expected Result

TECHNICAL INFORMATION ABOUT THE POKEMON VIDEO GAME

What is Pokemon?

Pokemon is a video game where players are required to raise a Pokemon from Level 1 to Level 99.

To reach a new level, each Pokemon must collect a certain number of **experience points**. The total number of experience points needed depends on the Pokemon's **growth rate type**.

You have been hired by Nintendo to test the system that calculates how many **experience points** are required for a Pokemon to reach the next level. A diagram of the overall system architecture is shown in ***Technical Information - Architecture Diagram*****.**

Background Information: Pokemon Growth Rate

Pokemon are classified into 1 of 3 growth rate categories:

- Fast growth rate
- Medium growth rate
- Slow growth rate

This means: A fast growth rate Pokemon will increase its level faster than a slow growth rate Pokemon

You can see which Pokemon belong in each category by clicking on the links below:

- Fast: https://bulbapedia.bulbagarden.net/wiki/Category:Pok%C3%A9mon_in_the_Fast_experience_group
- Medium: https://bulbapedia.bulbagarden.net/wiki/Category:Pok%C3%A9mon_in_the_Medium_Fast_experience_group
- Slow: https://bulbapedia.bulbagarden.net/wiki/Category:Pok%C3%A9mon_in_the_Slow_experience_group

Experience Points

Pokemon require a certain number of experience points to get to the next level.

Here is the formula for how many points are needed:

Pokemon Type	Formula
Fast	$0.8(L^3)$
Medium	L^3
Slow	$1.25(L^3)$

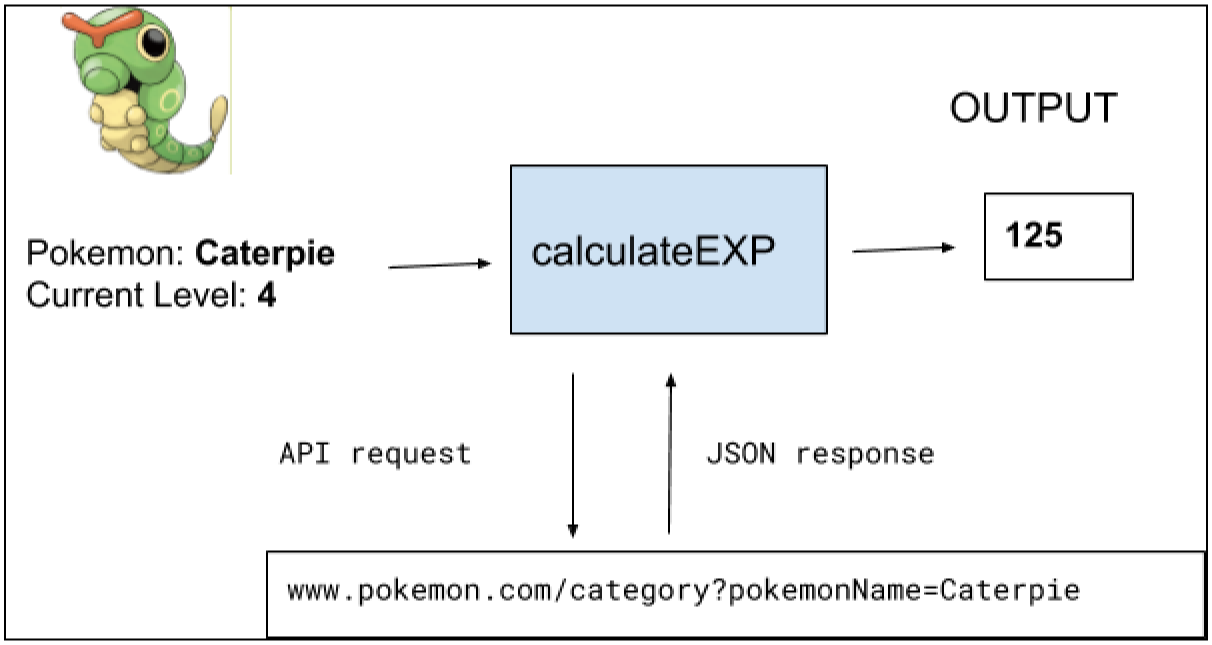
L = the level you want to reach

Example: for a **slow growth** pokemon, the pokemon needs:

- 33.75 points to reach level 3 ($1.25 \times 3^3 = 33.75$)
- 156 points to reach level 5 ($1.25 \times 5^3 = 156.25$)
- 640,000 points to reach level 80 ($1.25 \times 80^3 = 640,000$)

Technical Information - Architecture Diagram

The following is a system-level diagram of how the function works:



Function header and description of internal behaviour:

```
public int calculateExperiencePoints(String pokemonName, int currentLevel)
{
   // 1. Lookup the pokemon’s growth rate category via an API
   // 2. Calculate the number of points needed to get to next level
   // 3. Return the number of points required to reach next level
}
```

Technical Information - API Behaviour

Nintendo has an internal REST-API that is used to manage interactions with the Pokemon database. The API’s endpoints are as follows:

REQUEST	URL	QUERY PARAMETERS	EXPECTED BEHAVIOUR	EXPECTED RESPONSE CODE
GET	/pokemon		Returns all pokemon JSON response = Array of dictionaries Each dictionary contains the id, name, and color of Pokemon	200
GET	/pokemon	id	Returns information about a specific Pokemon JSON response = A single dictionary object containing the id, name, and color of a Pokemon	200
GET	/category		Returns all possible Pokemon growth rate categories JSON response = Array of strings.	200

			Each string is a growth rate category.	
GET	/category	pokemonName	Returns the growth rate category for the specified Pokemon JSON response = a dictionary object containing the id, name, growth rate category of the Pokemon	200

POST	/pokemon	pokemonName
		pokemonType
		color
		growthCategory
			Adds a new pokemon to the database
			
			JSON Response = an array containing a single dictionary. The dictionary looks like this:
			
			{
			  “id”:____
			}
			
			___ is the id of the newly created Pokemon.
			
			ids are automatically generated by the database.
				201

POST	/category		This endpoint is not implemented.	501

‘PUT	/pokemon	id
		pokemonName
		pokemonType
			Updates the Pokemon with the specified id
			
			JSON Response =
			{
			  “status”:”success”
			}
				201

PUT	/category	id growthCategory	Updates the growth rate category of the Pokemon with the specified id.	420

DELETE	/pokemon	id	Deletes the pokemon with the specified id
			
			JSON Response =
			{
			  “status”:”success”
			}
				200

DELETE	/category		This endpoint is not implemented	501