**IHUOMA FAVOUR AGBARU**

**WEEK 4 HOMEWORK**

**TASK 1 (Git and GitHub)**

**Question 1**

Complete definitions for key Git & GitHub terminology.

**GIT WORKFLOW FUNDAMENTALS**

· **Working Directory**

**Answer:** This is the area where the current working files lives and an untracked area of Git.

· **Staging Area:**

**Answer:** The staging area is where git starts tracking and saving changes that occurs in the files.

· **Local Repo (head)**

**Answer:** The local repository is the area that saves everything that has been committed using the git commit commands.

· **Remote repo (master)**

**Answer:** This is the uniform resource locator address of where git stores your project versions on the internet.

**WORKING DIRECTORY STATES:**

· **Staged**

**Answer:** This means that you have marked the changed file in its current version to go into your next commit snapshot.

· **Modified**

**Answer:** this means that you have modified the file but have not devoted it to your database yet.

· **Committed**

**Answer:** This means that the record/data is competently saved in your local database.

**GIT COMMANDS:**

· **Git add:**

This adds new or changes files in the working directory to the staging area.

· **Git commit**

**Answer:** The Git commit takes snapshots of your repository at every point

· **Git push**

**Answer:** The git push command pushes the local repository into the remote repository after changes has been made on the repository.

· **Git fetch**

**Answer:** This is when git gathers any commits from the target branch that does not exist in the current branch.

· **Git merge**

**Answer:** The git merge command merges the branches in the case where there is more than a branch, it merges all the branches and forks them into a single branch.

· **Git Pull**

**Answer:** This command fetches and merges changes from the remote server to the working directory.

**TASK 2 (Exception Handling)**

**Question 1**

**Simple ATM program**

Using exception handling code blocks such as try/ except / else / finally, write a

program that simulates an ATM machine to withdraw money.

(NB: the more code blocks the better, but try to use at least two key words e.g.

try/except)

**Tasks:**

1. Prompt user for a pin code

2. If the pin code is correct then proceed to the next step, otherwise ask a user to

type in a password again. You can give a user a maximum of 3 attempts and then

exit a program.

3. Set account balance to 100.

4. Now we need to simulate cash withdrawal

5. Accept the withdrawal amount

6. Subtract the amount from the account balance and display the remaining

balance (NOTE! The balance cannot be negative!)

7. However, when a user asks to ‘withdraw’ more money than they have on their

account, then you need to raise an error an exit the program.

**TASK 3 (Testing)**

**Question 1**

Use the Simple ATM program to write unit tests for your functions.

You are allowed to re-factor your function to ‘untangle’ some logic into smaller

blocks of code to make it easier to write tests.

Try to write at least 5 unit tests in total covering various cases.