**HOMEWORK WEEK 4**

(handout for students)

**TASK 1 (Git and GitHub)**

**Question 1**

Complete definitions for key Git & GitHub terminology

GIT WORKFLOW FUNDAMENTALS

·        **Working Directory**

This is where all of the files being edite at the current time are stored, it is also known as the Working Tree.

·        **Staging Area**

Also known as the index, this is where all of the changes made in the form of commits are stored, waiting to be committed to the repository.

·        **Local Repo (head)**

This is the version of a repository that is stored on an individual’s computer and therefore only editable by them. It may or may not linked to any online repository.

·        **Remote repo (master)**

This is the repository that is stored online and may or may not be linked to a local repository. Remote repositories can be accessed by multiple people.

WORKING DIRECTORY STATES:

·        **Staged**

This state is when a file has been edited and marked to be committed to the repository. It is the stepping stone between the modified and committed stages.

·        **Modified**

This state is when a file has been modified but not marked to be committed to the repository yet

·        **Committed**

This is when the modified file has been successfully saved inside the specified repository.

GIT COMMANDS:

·        **Git add**

This command will add a file to the staging area, using ‘git add .’ will add multiple files to the staging area.

·        **Git commit**

This command saves the edited file into the version history.

·        **Git push**

This command sends the changes you have committed to the remote repository.

·        **Git fetch**

This command will retrieve commits made on the remote server but will not integrate it into the working directory.

·        **Git merge**

This command integrates all commits made in current branch into the master branch.

·        **Git pull**

This command fetches and merges the changes from the remote server into 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.