Foundation Assessment Theory: Minka Deborah Ansa

Q1.1: What does SDLC stand for?

SDLC stands for the Software Development Life Cycle

Q1.2: What exception is thrown when you divide a number by 0?

ZeroDivisionError is the name of the exception thrown when you try to divide a number by 0

Q1.3: What is the git command that moves code from the local repository to the remote repository?

git push is the git command that moves code from the local repository to the remote repository

Q1.4: What does NULL represent in a database?

NULL represents data that doesn’t exist in a database

Q1.5: Name 2 responsibilities of the Scrum Master

The Scrum Master facilitates daily stand-up meetings, and sprint reviews and sprint retrospectives

The scrum master is also responsible for meeting with the Development Team and Product Owner to select high priority items from the product backlog that can be delivered in a single sprint

Q1.6: Name 2 debugging methods, and when you would use them.

Print/Console.log() is one method of debugging. When you want your code to perform a certain task, you can add a console log/print statement at certain points that will provide certain values if triggered. You can use these to pinpoint problem areas in the code

Regression testing is another method of debugging. You would perform a regression test after any changes, like bug fixes, are made to ensure that these changes haven't caused another bug elsewhere.

Q1.7: Looking at the following code, describe a case where this function would throw an error when called.

Describe this case and talk about what exception handling you'll need.

```
def can_pay(price, cash_given):
   if cash_given >= price:
       return True
   else:
       return False
```

This code could raise a TypeError if the parameters given are not integers when this function is called because you cannot compare two different data types using >= operator.

A way to handle this error is to convert the arguments to a numeric value using float(). Then create an exception using a ValueError.

(We should use a ValueError instead of a TypeError because the float() function can take an integer value or a string that can be converted to an integer e.g "21" as opposed to "twenty one".)

Now if one of the parameters given is invalid e.g. "dog", rather than raising a ValueError, you can return a message instead:

```
def can_pay(price, cash_given):
    try:
        price = float(price)  # Converting price to a numeric value
        cash_given = float(cash_given)  # Converting cash_given to a numeric value
    except ValueError:  # In case the price or cash_given is not a valid numeric value
        print('Error: Please enter a numerical value')  # Rather than raising a ValueError

    if cash_given >= price:
        print (True)
    else:
```

```
        print (False)


can_pay("twenty", "thirty")
# Error: Please enter a numerical value

can_pay("20","30")
# True

can_pay(20, 30)
# True
```

Q1.8: What is git branching? Explain how it is used in Git.

git branching allows developers to code collaboratively without disrupting the main file. In git, you create a new file using git branch, make changes to that code in that branch and then submit these changes for approval using git add, git commit and git push.

Q1.9: Design a restaurant ordering system.

You do not need to write code, but describe a high-level approach:
a. Draw a list of key requirements
b. What are your main considerations and problems?
c. What components or tools would you potentially use?

Key features:

Phone/Desktop App

Ability for users to log in

User Home/Landing Page

Menu management defining the different menus including prices, descriptions, and pictures of each item

Order management for user to add items to basket and place orders

Payment processing supporting methods like apple and google pay as well as credit/debit cards, as well as the ability to store payment details

Order dashboard for restaurants to see and confirm orders placed as well as update menu prices and stock

Delivery integration including location services to allow in-house or third-party order delivery

Recommendations orders to users based on statistics

e.g. Top ordered in local area, Least expensive, Closest distance to you etc…

Accessibility options like dark mode, high contrast etc..

Customer/Help Centre for wrong/incorrect orders

Common problems:

Scalability; the app should be capable of handling various users during peak times without compromised performance. If you get logged out, lose connection, or the app crashes is your order still processed?

Security; the app should be able to store user data. How are payment details or addresses kept secure? How is location data handled safely?

Localisation; the app should be able to handle multiple currencies, languages and timezones.

User experience; the app should be accessible, intuitive and responsive. How many addresses can you have per user? How is the driver's location synced to the user's profile during delivery? How quickly are orders processed? How easy is it for the restaurants to update their menu/prices?

Architecture to consider:

HTML, CSS & JS for desktop site, Swift for iOS app, Kotlin for Android app

Front end: web or mobile application for customers to place orders and a separate dashboard or app for restaurant staff to manage orders and menus

Back end: server to handle user authentication, order processing, and communication with external services.

Database: for storing user data, menus, orders, and transaction records.

Payment integrations like Stripe or PayPal

Security features for authentication and user privacy

APIs

Location Services: for order tracking and delivery services