Reg. No. 2447116
Name: Chloy costa

Q1. You are lost in a jungle and have a list of landmarks to follow to find your way out. However, some landmarks are in the correct order while others are jumbled. (5 Marks)
Given the following lists:
i. Ordered Part: Some landmarks in the correct order
ordered_part = ['Start', 'River']
ii. Jumbled Part: The rest of the landmarks in a random order.
jumbled_part = ['Clearing', 'Village', 'Cave']

The correct order of all landmarks is:
• 'Start'
• 'Clearing'
• 'River'
• 'Village'
• 'Cave'
Write a Python function reconstruct_path that combines these given lists to form the complete ordered path.

Code:

```
ordered_part = ["Start", "River"]

jumbled_part = ["Clearing", "Village", "Cave"]
corrected_order = []

def reconstruct_path(ordered, jumbled):
   correct_order = ["Start","Clearing", "River", "Village", "Cave" ]
   for i in correct_order:
       if i in ordered:
           corrected_order.append(i)
       elif i in jumbled:
           corrected_order.append(i)

   return corrected_order

print(reconstruct_path(ordered_part, jumbled_part))
```

Output :

Q2. Create a class called "Book" that inherits from "LibraryItem" and includes the following attributes and methods: (12 Marks)
• Additional Attributes: author, genre, isbn
• Methods:
o Checkout() - which sets the availability to False and prints "Book checked out"
(2 Marks)
o ReturnBook() - which sets the availability to True and prints "Book returned"
(2 Marks)
o UpdateAvailability(new_status) - which updates the availability status of the book. If the book is not available, print "Book is currently checked out".

(2 Marks)

o GetDetails() - which overrides the method to print the details of the book including title, author, genre, and isbn.

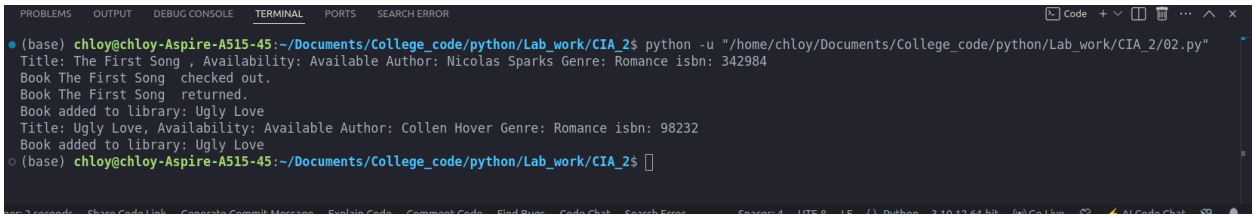
(2 Marks)
o AddToLibrary() - which prints a message saying "Book added to library: {title}"
(2 Marks)
• Create two objects of the Book class and call the functions.

Code:

```
class Library_item:
    def __init__(self, title , availability = True):
        self.title = title
        self.availability = availability

    def get_details(self):
        return f"Title: {self.title}, Availability: {'Available' if
self.availability else 'Not Available'}"

class Book(Library_item):
    def __init__(self, title, author, genre, isbn, availability = True):
        self.author = author
        self.genre = genre
        self.isbn = isbn
```

```
        super().__init__(title, availability)

    def checkout(self):
        if self.availability:
            self.availability = False
            return f"Book {self.title} checked out."
        else:
            return f"Book {self.title} is not available, it has already
been checked out."

    def return_book(self):
        self.availability = True
        return f"Book {self.title} returned."

    def update_availability(self, new_status):
        if not new_status and not self.availability:
            print("Book is currently checked out")
        else:
            self.availability = new_status

    def get_details(self):
        details = super().get_details()
        return (f"{details} Author: {self.author} Genre: {self.genre} isbn:
{self.isbn}")

    def add_to_library(self):
        return (f"Book added to library: {self.title}")

book01 = Book("The First Song ", "Nicolas Sparks", "Romance", "342984")
book02 = Book("Ugly Love", "Collen Hover", "Romance", "98232")

print(book01.get_details())
print(book01.checkout())
print(book01.return_book())

print(book02.add_to_library())
print(book02.get_details())
print(book02.add_to_library())
```

Output:



Q3. Implement the concept of Inheritance and Polymorphism: (8 Marks)
• Create a base class "LibraryItem" with the get_details() method.

(3 Marks)
• Create two derived classes "Magazine" and "DVD" that inherit from "LibraryItem" and implement the get_details() method. Add specific attributes like issue_number for Magazine and duration for DVD.

(3 Marks)
• Create objects of both classes and call their methods.

Code:

```
class Library_item:
    def __init__(self, title , availability = True):
        self.title = title
        self.availability = availability

    def get_details(self):
        return f"Title: {self.title}, Availability: {'Available' if
self.availability else 'Not Available'}"

class Magazine(Library_item):
    def __init__(self, title, issue_number, availability=True):
        super().__init__(title, availability)
        self.issue_number = issue_number

    def get_details(self):
        details = super().get_details()
        return f"{details}, Issue Number: {self.issue_number}"
```

```python
class DVD(Library_item):
    def __init__(self, title, duration, availability=True):
        super().__init__(title, availability)
        self.duration = duration

    def get_details(self):
        details = super().get_details()
        return f"{details} Duration: {self.duration}"

magazine = Magazine("Times of India", "938236")

dvd = DVD("Tarzen", "25mins")

print(magazine.get_details())
print(dvd.get_details())
```

Output: