ASSIGNMENT 6.5

NAME: Karthik Reddy
HT NO: 2303A51335
BATCH: 20

Task Description #1 (AI-Based Code Completion for Conditional Eligibility Check)
Task: Use an AI tool to generate eligibility logic.
Prompt:
“Generate Python code to check voting eligibility based on age and citizenship.”
Expected Output:
• AI-generated conditional logic.
• Correct eligibility decisions.
• Explanation of conditions.
CODE:

```python
#Generate Python code to check voting eligibility based on age and citizenship
age = int(input("Enter your age: "))
citizenship = input("Are you a citizen? (yes/no): ").strip().lower()
if age >= 18 and citizenship == 'yes':
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
# This code prompts the user to enter their age and citizenship status.
# It then checks if the user is 18 years or older and a citizen to determine voting eligibility.
```

Output:

```
/OneDrive/Desktop/Ai asst Coding/assignment 6.5.py"
Enter your age: 19
Are you a citizen? (yes/no): yes
You are eligible to vote.
```

Task Description #2(AI-Based Code Completion for Loop-Based String Processing)
Task: Use an AI tool to process strings using loops.
Prompt:
“Generate Python code to count vowels and consonants in a string using a loop.”
Expected Output:
• AI-generated string processing logic.
• Correct counts.
• Output verification.

CODE:

```python
#Generate Python code to count vowels and consonants in a string using a loop.”
text = input("Enter a string: ").lower()
vowels = "aeiou"
vowel_count = 0
consonant_count = 0
for char in text:
    if char.isalpha():
        if char in vowels:
            vowel_count += 1
        else:
            consonant_count += 1
print(f"Vowels: {vowel_count}, Consonants: {consonant_count}")
# This code takes a string input from the user and counts the number of vowels and consonants
# by iterating through each character in the string.
# It uses conditional statements to check if a character is a vowel or consonant.
```

OUTPUT:

```
Enter a string: recommended
Vowels: 4, Consonants: 7
```

Task Description #3 (AI-Assisted Code Completion Reflection Task)
Task: Use an AI tool to generate a complete program using classes, loops, and conditionals.
Prompt:
“Generate a Python program for a library management system using classes, loops, and conditional statements.”
Expected Output:
• Complete AI-generated program.
• Review of AI suggestions quality.
• Short reflection on AI-assisted coding experience.

CODE:

```python
#Generate a Python program for a library management system using classes, loops, and conditional statements
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_available = True
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book):
        self.books.append(book)
    def display_books(self):
        for book in self.books:
            status = "Available" if book.is_available else "Checked Out"
            print(f"{book.title} by {book.author} - {status}")
    def checkout_book(self, title):
        for book in self.books:
            if book.title == title and book.is_available:
                book.is_available = False
                print(f"You have checked out '{book.title}'.")
                return
        print(f"Sorry, '{title}' is not available.")
    def return_book(self, title):
        for book in self.books:
            if book.title == title and not book.is_available:
                book.is_available = True
                print(f"You have returned '{book.title}'.")
                return
        print(f"'{title}' was not checked out.")
library = Library()
library.add_book(Book("1984", "George Orwell"))
```

```python
library.add_book(Book("1984", "George Orwell"))
library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
while True:
    print("\nLibrary Menu:")
    print("1. Display Books")
    print("2. Checkout Book")
    print("3. Return Book")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")
    if choice == '1':
        library.display_books()
    elif choice == '2':
        title = input("Enter the title of the book to checkout: ")
        library.checkout_book(title)
    elif choice == '3':
        title = input("Enter the title of the book to return: ")
        library.return_book(title)
    elif choice == '4':
        print("Exiting the library system.")
        break
    else:
        print("Invalid choice. Please try again.")
# This code defines a simple library management system using classes.
# The Book class represents a book with its title, author, and availability status.
# The Library class manages a collection of books and provides methods to add, display, checkout, and retu
# A loop is used to present a menu to the user for interacting with the library system.
# The program continues until the user chooses to exit.
# The program uses conditional statements to handle user choices and book availability.
```

OUTPUT:

```
Library Menu:
1. Display Books
2. Checkout Book
3. Return Book
4. Exit
Enter your choice (1-4): 1
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
The Great Gatsby by F. Scott Fitzgerald - Available

Library Menu:
1. Display Books
2. Checkout Book
3. Return Book
4. Exit
Enter your choice (1-4): 2
Enter the title of the book to checkout: The Great Gatsby
You have checked out 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Checkout Book
3. Return Book
4. Exit
```

```
Library Menu:
1. Display Books
2. Checkout Book
3. Return Book
4. Exit
Enter your choice (1-4): 4
```

Task Description #4 (AI-Assisted Code Completion for Class- Based Attendance System)

Task: Use an AI tool to generate an attendance management class.

Prompt: “Generate a Python class to mark and display student attendance using loops.”

Expected Output:

- AI-generated attendance logic.
- Correct display of attendance.
- Test cases.

CODE:

```
#Generate a Python class to mark and display student attendance using loops.
class Student:
    def __init__(self, name):
        self.name = name
        self.attendance = []
    def mark_attendance(self, status):
        self.attendance.append(status)
    def display_attendance(self):
        present_days = self.attendance.count('Present')
        total_days = len(self.attendance)
        print(f"{self.name}'s Attendance: {present_days}/{total_days} days present.")
students = []
num_students = int(input("Enter number of students: "))
for _ in range(num_students):
    name = input("Enter student name: ")
    students.append(Student(name))
num_days = int(input("Enter number of days to mark attendance: "))
for day in range(num_days):
    print(f"\nDay {day + 1} Attendance:")
    for student in students:
        status = input(f"Is {student.name} present? (yes/no): ").strip().lower()
        if status == 'yes':
            student.mark_attendance('Present')
        else:
            student.mark_attendance('Absent')
for student in students:
        else:
            student.mark_attendance('Absent')
for student in students:
    student.display_attendance()
# This code defines a Student class to manage attendance records.
# It allows marking attendance for multiple students over several days using loops.
# The program prompts the user to enter student names and their attendance status for each day.
# Finally, it displays the attendance summary for each student.
```

OUTPUT:

```
Enter number of students: 3
Enter student name: ram
Enter student name: kiran
Enter student name: raju
Enter number of days to mark attendance: 4

Day 1 Attendance:
Is ram present? (yes/no): yes
Is kiran present? (yes/no): no
Is raju present? (yes/no): no

Day 2 Attendance:
Is ram present? (yes/no): yes
Is kiran present? (yes/no): yes
Is raju present? (yes/no): yes

Day 3 Attendance:
Is ram present? (yes/no): no
Is kiran present? (yes/no): no
Is raju present? (yes/no): yes

Day 4 Attendance:
Is ram present? (yes/no): no
Is kiran present? (yes/no): yes
Is raju present? (yes/no): no
ram's Attendance: 2/4 days present.
kiran's Attendance: 2/4 days present.
raju's Attendance: 2/4 days present.
```

Task Description #5 (AI-Based Code Completion for Conditional Menu Navigation)
Task: Use an AI tool to complete a navigation menu.
Prompt: “Generate a Python program using loops and conditionals to simulate an ATM menu.”
Expected Output:
• AI-generated menu logic.
• Correct option handling.
• Output verification.

CODE:

```python
#Generate a Python program using loops and conditionals to simulate an ATM menu.”
class ATM:
    def __init__(self, balance=0):
        self.balance = balance
    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited: ${amount}. New balance: ${self.balance}.")
    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds.")
        else:
            self.balance -= amount
            print(f"Withdrew: ${amount}. New balance: ${self.balance}.")
    def check_balance(self):
        print(f"Current balance: ${self.balance}.")
atm = ATM(1000)  # Initial balance of $1000
while True:
    print("\nATM Menu:")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Check Balance")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")
    if choice == '1':
        amount = float(input("Enter amount to deposit: "))
        atm.deposit(amount)
    elif choice == '2':
        amount = float(input("Enter amount to withdraw: "))
        atm.withdraw(amount)
    elif choice == '3':
        atm.check_balance()
```

```python
        atm.withdraw(amount)
    elif choice == '3':
        atm.check_balance()
    elif choice == '4':
        print("Exiting ATM. Thank you!")
        break
    else:
        print("Invalid choice. Please try again.")
# This code defines an ATM class with methods for depositing, withdrawing, and checking balance.
# A loop presents a menu to the user for interacting with the ATM system.
# The program continues until the user chooses to exit.
# Conditional statements handle user choices and ensure valid operations.
```

OUTPUT:

```
ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 1
Enter amount to deposit: 199999
Deposited: $199999.0. New balance: $200999.0.

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 2
Enter amount to withdraw: 4000
Withdrew: $4000.0. New balance: $196999.0.

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 3
Current balance: $196999.0.

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 4
Exiting ATM. Thank you!
```