TASK 1:

# Function to calculate the check digit for the frequent parking number

def calculate\_check\_digit(number):

    total = 0

    for i in range(len(number)):

        total += int(number[i]) \* (i + 1)

    return total % 11

# Function to calculate the parking price

def calculate\_price(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

    discounts = {

        'Sunday': 0.1,

        'Monday': 0.1,

        'Tuesday': 0.1,

        'Wednesday': 0.1,

        'Thursday': 0.1,

        'Friday': 0.1,

        'Saturday': 0.5

    }

    hour\_price = 2 if arrival\_hour < 16 else 3  # Hourly price based on arrival time

    discount = discounts[day\_of\_week] if frequent\_parking\_number[-1] == str(calculate\_check\_digit(frequent\_parking\_number[:-1])) else 0.0  # Check discount

    total\_price = hour\_price \* hours\_parked \* (1 - discount)  # Calculate total price

    return total\_price

# Function to validate input data

def validate\_input(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

    # Add validation logic here

    return True  # For simplicity, assuming all inputs are valid

# Main program

def main():

    day\_of\_week = input("Enter the day of the week: ")

    arrival\_hour = int(input("Enter the hour of arrival (0-23): "))

    hours\_parked = int(input("Enter the number of hours parked: "))

    frequent\_parking\_number = input("Enter the frequent parking number (5 digits): ")

    if validate\_input(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

        price = calculate\_price(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number)

        print("The price to park is: $", round(price, 2))

    else:

        print("Invalid input. Please try again.")

main()  # Call the main function directly

TASK 2:

# Function to calculate the check digit for the frequent parking number

def calculate\_check\_digit(number):

    total = 0

    for i in range(len(number)):

        total += int(number[i]) \* (i + 1)

    return total % 11

# Function to calculate the parking price

def calculate\_price(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

    discounts = {

        'Sunday': 0.1,

        'Monday': 0.1,

        'Tuesday': 0.1,

        'Wednesday': 0.1,

        'Thursday': 0.1,

        'Friday': 0.1,

        'Saturday': 0.5

    }

    hour\_price = 2 if arrival\_hour < 16 else 3  # Hourly price based on arrival time

    discount = discounts[day\_of\_week] if frequent\_parking\_number[-1] == str(calculate\_check\_digit(frequent\_parking\_number[:-1])) else 0.0  # Check discount

    total\_price = hour\_price \* hours\_parked \* (1 - discount)  # Calculate total price

    return total\_price

# Function to validate input data

def validate\_input(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

    # Add validation logic here

    return True  # For simplicity, assuming all inputs are valid

# Main program

def main():

    daily\_total\_payments = 0  # Initialize daily total payments

    while True:

        total\_payments = 0  # Initialize total payments for the current transaction

        # Input and calculation for Task 1

        day\_of\_week = input("Enter the day of the week (or type 'end' to finish): ")

        if day\_of\_week.lower() == 'end':

            break  # End the program if the user enters 'end'

        arrival\_hour = int(input("Enter the hour of arrival (0-23): "))

        hours\_parked = int(input("Enter the number of hours parked: "))

        frequent\_parking\_number = input("Enter the frequent parking number (5 digits): ")

        if validate\_input(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

            price = calculate\_price(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number)

            print("The price to park is: $", round(price, 2))

            total\_payments += price  # Add the payment to the total payments for this transaction

            daily\_total\_payments += total\_payments  # Add the payment to the daily total

        else:

            print("Invalid input. Please try again.")

    # Display total payments for the day

    print("Total payments for the day: $", round(daily\_total\_payments, 2))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

TASK 3:

# Function to calculate the check digit for the frequent parking number

def calculate\_check\_digit(number):

    total = 0

    for i in range(len(number)):

        total += int(number[i]) \* (i + 1)

    return total % 11

# Function to calculate the parking price (including Task 3 modifications)

def calculate\_price(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

    discounts = {

        'Sunday': 0.1,

        'Monday': 0.1,

        'Tuesday': 0.1,

        'Wednesday': 0.1,

        'Thursday': 0.1,

        'Friday': 0.1,

        'Saturday': 0.5

    }

    # Hourly price based on arrival time

    hour\_price\_before\_4pm = 2  # Regular price before 16:00

    hour\_price\_after\_4pm = 3  # Regular price after 16:00

    # Calculate price before 16:00

    price\_before\_4pm = hour\_price\_before\_4pm \* min(hours\_parked, 16 - arrival\_hour) \* (1 - discounts[day\_of\_week])

    # Calculate price after 16:00 (if applicable)

    if arrival\_hour < 16 and hours\_parked > (16 - arrival\_hour):

        price\_after\_4pm = hour\_price\_after\_4pm \* (hours\_parked - (16 - arrival\_hour))

    else:

        price\_after\_4pm = 0

    total\_price = price\_before\_4pm + price\_after\_4pm  # Calculate total price

    return price\_before\_4pm, price\_after\_4pm

# Placeholder function to validate input data

def validate\_input(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

    # Placeholder validation logic (always returns True)

    return True

# Main program

def main():

    daily\_total\_payments\_before\_4pm = {}  # Initialize daily total payments before 4pm dictionary

    daily\_total\_payments\_after\_4pm = {}  # Initialize daily total payments after 4pm dictionary

    while True:

        # Input and calculation for Task 1

        day\_of\_week = input("Enter the day of the week (or type 'end' to finish): ")

        if day\_of\_week.lower() == 'end':

            break  # End the program if the user enters 'end'

        arrival\_hour = int(input("Enter the hour of arrival (0-23): "))

        hours\_parked = int(input("Enter the number of hours parked: "))

        frequent\_parking\_number = input("Enter the frequent parking number (5 digits): ")

        if validate\_input(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number):

            price\_before\_4pm, price\_after\_4pm = calculate\_price(day\_of\_week, arrival\_hour, hours\_parked, frequent\_parking\_number)

            print(f"Price before 16:00 on {day\_of\_week}: ${round(price\_before\_4pm, 2)}")

            print(f"Price after 16:00 on {day\_of\_week}: ${round(price\_after\_4pm, 2)}")

            # Add the payment to the total payments for this day (before 4pm)

            if day\_of\_week in daily\_total\_payments\_before\_4pm:

                daily\_total\_payments\_before\_4pm[day\_of\_week] += price\_before\_4pm

            else:

                daily\_total\_payments\_before\_4pm[day\_of\_week] = price\_before\_4pm

            # Add the payment to the total payments for this day (after 4pm)

            if day\_of\_week in daily\_total\_payments\_after\_4pm:

                daily\_total\_payments\_after\_4pm[day\_of\_week] += price\_after\_4pm

            else:

                daily\_total\_payments\_after\_4pm[day\_of\_week] = price\_after\_4pm

        else:

            print("Invalid input. Please try again.")

    # Display total payments for each day before 4pm

    print("\nTotal payments before 16:00 for each day:")

    for day, total in daily\_total\_payments\_before\_4pm.items():

        print(f"{day}: ${round(total, 2)}")

    # Display total payments for each day after 4pm

    print("\nTotal payments after 16:00 for each day:")

    for day, total in daily\_total\_payments\_after\_4pm.items():

        print(f"{day}: ${round(total, 2)}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()