3. WRITTEN RESPONSES

3 a.

3.a.i.

The overall purpose of the program is for a user to get mortgage amortization schedule including monthly payment, total interest and estimated payoff date. The schedule also shows how much money a user needs to pay in principal and interest over the period of loan term.

3.a.ii.

In the video, it shows an example of how the user will input data required to make the schedule, and then views the schedule from the saved text file.

3.a.iii.

The inputs demonstrated in the videos are the mortgage amount, mortgage term, interest rate, start month and year, mode of schedule output, and extra payments. The output shown is the text file which contains the amortization schedule.

3 b.

3.b.i.

```
amortization_schedule = [
    ["Month", "Year", "Monthly Payment", "Interest", "Principal", "Total Interest Paid", "Balance"],
]
```

3.b.ii.

```
def Create_amrt_schedule(Balance_Remaining, Interest_Rate, Monthly_Payment, Total_Interest_Paid, Month, Year,
                         amrt_output):
    Extra1, Extra2, Extra2_Month, Extra3, Extra3_Month, Extra3_Year = Extras()
    while Balance_Remaining > 0.1:
        Monthly_Interest = Monthly_Interest_Function(Balance_Remaining, Interest_Rate)
        Total_Interest_Paid = Total_Interest_Paid_Function(Total_Interest_Paid, Monthly_Interest)
        if Monthly_Payment > Balance_Remaining:
            Monthly_Principal = Balance_Remaining
            Monthly_Payment = Monthly_Principal + Monthly_Interest
        else:
            Monthly_Principal = Monthly_Principal_Function(Monthly_Payment, Monthly_Interest, Month, Year, Extra1,
                                                           Extra2,
                                                           Extra2_Month, Extra3, Extra3_Month, Extra3_Year)
        Balance_Remaining = Monthly_Balance_Remaining_Function(Balance_Remaining, Monthly_Principal)

        amortization_schedule.append(
            [Months_Of_The_Year[Month - 1], Year, round(Monthly_Payment, 2), round(Monthly_Interest, 2),
             round(Monthly_Principal, 2), round(Total_Interest_Paid, 2), round(Balance_Remaining, 2)]
        )
        if Month >= 12:
            Month = 0
            Year += 1
        Month += 1
    print("Your monthly payment amount is:", round(Monthly_Payment,2), "dollars")
    print("Your total interest to be paid is", round(Total_Interest_Paid,2), "dollars")
    print("Your estimated payoff date is", Months_Of_The_Year[Month - 2], Year)
    table = AsciiTable(amortization_schedule)
    if amrt_output == 'VIEW':
        print(table.table)
    elif amrt_output == 'SAVE':
        f = open(Name_File + ".txt", "w")
        f.write(table.table)
    else:
        f = open(Name_File + ".txt", "w")
        f.write(table.table)
        print(table.table)
```

3.b.iii.

The name of the list is “amortization_schedule”.

3.b.iv.

The data in the “amortization_schedule” list represents the amortization schedule for the mortgage payment. It contains the month and year, monthly payment, principal, interest, total interest paid, and balance remaining.

3.b.v.

The list “amortization_schedule” manages complexity in my program code by keeping the entire amortization schedule neat and organized, and also, makes it easier to print and/or save the schedule by using it as a list. If I did not use the list, I would have to print and save the data for every month line by line, which will slow down the program and make the code cluttered and difficult to maintain.

3 c.

3.c.i.

```
def Create_amrt_schedule(Balance_Remaining, Interest_Rate, Monthly_Payment, Total_Interest_Paid, Month, Year,
                         amrt_output):
    Extra1, Extra2, Extra2_Month, Extra3, Extra3_Month, Extra3_Year = Extras()
    while Balance_Remaining > 0.1:
        Monthly_Interest = Monthly_Interest_Function(Balance_Remaining, Interest_Rate)
        Total_Interest_Paid = Total_Interest_Paid_Function(Total_Interest_Paid, Monthly_Interest)
        if Monthly_Payment > Balance_Remaining:
            Monthly_Principal = Balance_Remaining
            Monthly_Payment = Monthly_Principal + Monthly_Interest
        else:
            Monthly_Principal = Monthly_Principal_Function(Monthly_Payment, Monthly_Interest, Month, Year, Extra1,
                                                           Extra2,
                                                           Extra2_Month, Extra3, Extra3_Month, Extra3_Year)
        Balance_Remaining = Monthly_Balance_Remaining_Function(Balance_Remaining, Monthly_Principal)

        amortization_schedule.append(
            [Months_Of_The_Year[Month - 1], Year, round(Monthly_Payment, 2), round(Monthly_Interest, 2),
             round(Monthly_Principal, 2), round(Total_Interest_Paid, 2), round(Balance_Remaining, 2)]
        )
        if Month >= 12:
            Month = 0
            Year += 1
        Month += 1
    print("Your monthly payment amount is:", round(Monthly_Payment,2), "dollars")
    print("Your total interest to be paid is", round(Total_Interest_Paid,2), "dollars")
    print("Your estimated payoff date is", Months_Of_The_Year[Month - 2], Year)
    table = AsciiTable(amortization_schedule)
    if amrt_output == 'VIEW':
        print(table.table)
    elif amrt_output == 'SAVE':
        f = open(Name_File + ".txt", "w")
        f.write(table.table)
    else:
        f = open(Name_File + ".txt", "w")
        f.write(table.table)
        print(table.table)
```

3.c.ii.

```
def main():
    global Name_File
    Mortgage_Amount = float(input("Enter the mortgage amount: "))
    Mortgage_Term = (float(input("Enter the mortgage term in years: "))) * 12
    Interest_Rate = (float(input("Enter the annual interest rate percentage (numbers only): "))) / (12 * 100)

    Month = int(input("Enter the month number between 1 to 12 from when your mortgage will start: "))
    Year = int(input('Enter the year from when your mortgage will start: '))

    Monthly_Payment = Monthly_Payment_Function(Mortgage_Amount, Interest_Rate, Mortgage_Term)

    Balance_Remaining = Mortgage_Amount
    Total_Interest_Paid = 0

    print("Do you want to view the amortization schedule, save it in a text file, or do both?")
    Schedule_type = int(input("1 = view, 2 = save, 3 = both \nChoose one: "))
    if Schedule_type == 1:
        Create_amrt_schedule(Balance_Remaining, Interest_Rate, Monthly_Payment, Total_Interest_Paid, Month, Year,
                             "VIEW")
    elif Schedule_type == 2:
        Name_File = input("Type the name of the file you want to save the schedule as: ")
        Create_amrt_schedule(Balance_Remaining, Interest_Rate, Monthly_Payment, Total_Interest_Paid, Month, Year,
                             "SAVE")
    elif Schedule_type == 3:
        Name_File = input("Type the name of the file you want to save the schedule as: ")
        Create_amrt_schedule(Balance_Remaining, Interest_Rate, Monthly_Payment, Total_Interest_Paid, Month, Year,
                             "BOTH")
```

3.c.iii.

The procedure “Create_amrt_schedule” is used to determine the entire amortization schedule by calculating monthly interest, principal, remaining balance, total interest paid and estimated payoff date. It also calls another function to capture extra payments to be made by the user and then adjusts principal accordingly. Lastly, it prints and/or saves the amortization schedule.

3.c.iv.

The procedure “Create_armt_schedule” first calls another procedure to capture if the user wants to add any extra payments. Then, a while loop is initiated to add monthly data to the list “amortization_schedule” until remaining balance is zero. Within the while loop, it calls the procedure “Monthly_Interest_Function” to calculate monthly interest and “Total_Interest_Paid_Function” to calculate how much interest has been paid off so far. Then, if the Monthly Payment is greater than the balance remaining, the monthly principal will be set equal to the balance remaining, and the monthly payment will be set equal to the monthly principal plus monthly interest. If the Monthly Payment is less than or equal to the balance remaining, then the “Monthly_Principal_Function” is called to calculate monthly principal. The next procedure being called is “Monthly_Balance_Remaining_Function”, which is used to calculate remaining balance. Every time the monthly data is calculated, it is added to “amortization_schedule”. It also increases month counter to move to the next month within same year or reset month to ‘January’ of next year. Then, it prints monthly payment, total interest, and estimated payoff date. After that, it converts “amortization_schedule” into a table using the imported AsciiTable function. Lastly, it prints and/or saves the entire amortization schedule depending on the user selection.

3 d.

3.d.i.

First call:

The first call is located inside the procedure “main”. It will be called when the user decides to view the amortization schedule in the terminal. It will pass the normal parameters like balance remaining, interest rate, monthly payment, total interest paid, month, and year. In addition, it will also pass a parameter called 'VIEW', which triggers a selection statement inside the procedure "Create_armt_schedule" in order to print the amortization schedule in the terminal.

Second call:

The second call is located inside the procedure “main”. It will be called when the user decides to save the amortization schedule as a text file. It will pass the normal parameters like balance remaining, interest rate, monthly payment, total interest paid, month, and year. In addition, it will also pass a parameter called 'SAVE', which triggers a selection statement inside the procedure "Create_armt_schedule" in order to save the amortization schedule as a text file.

3 d.ii.

Condition(s) tested by first call:

The condition tested in the first call will be whether or not the schedule prints into the terminal.

Condition(s) tested by second call:

The condition testing in the second call will be whether or not the schedule is saved as a text file.

3.d.iii.

Results of the first call:

The result of the first call is that the schedule is printed into the terminal.

Results of the second call:

The result of the second call is that the schedule is saved as a text file.