Foundation Certificate in Higher Education

Module Code: DOC334

Module Title: Computer Programming

Module Leader: Mr. Nishan Saliya Harankahawa

Assignment Number: 1

Assignment Type: Individual

Student ID: 20222153

Student Name: D.W.A. Nawoda Nethmini Wadisingha

Acknowledgements

Table of Contents

List of Figures

List of Tables

1. Problem Specification (Introduction)

The problem statement involves developing a console application in Python for a fictional country called "The Kingdom of Miranda." This country has five main cities, and its official currency is KMD (Kingdom of Miranda Dollars). The application is for a single cab service called DropMe™, which offers transportation between these cities using trishaws, cars, and vans.

The application must handle various commands issued by passengers to view trip details and generate invoices. Passengers can specify their preferred vehicle and may use promo codes for price reductions. The promo codes can be provided by the application or issued individually under special circumstances.

The application will calculate the fare based on the distance between the specified cities and the chosen vehicle type, taking the shortest path for travel. For cars, the fare is double, and for vans, it is triple the trishaw fare. Passengers may also receive random auto-generated promotions for a fixed amount of 5 KMD, but they won't be applied if the passenger uses a promo code.

The invoice for each trip is saved in a text file with the date, time, start, end, fare amount, promo code reduction (if any), random reduction (if any), and the final payment amount. Passengers can request to see the full price plan for all three vehicle types across the country.

The overall goal is to create an efficient and user-friendly application that allows passengers to check prices, apply promo codes, and receive invoices for their cab rides within The Kingdom of Miranda.

2. Developed Solution

2.1. Solution for the console application in "The Kingdom of Miranda"

1. Create dictionaries to store the price_list, vPrices, and promo codes with their corresponding values.

2. Define the get_initial_price function:
- Input: price_list, start_place, end_place
- Output: initial_price
- Get the price from the price_list dictionary based on the start and end cities.
- Return the initial_price.

3. Define the create_invoice function:
- Input: start, end, final_payment, promo_reduction, random_reduction
- Output: None
- Get the current date and time.
- Format the datetime to create a unique filename.
- Create a text file with the formatted filename.
- Write the trip details, including the date, time, start city, end city, fare amount, promo code reduction (if any), random reduction (if any), and the final payment amount, to the text file.
- Close the file.

4. Define the main function:
- Input: Command-line arguments args
- Output: None
- Parse the command-line arguments to identify the user commands and parameters.
- If the command is to show the full price plan for all vehicle types (dm /price):
- Loop through the price_list dictionary and multiply the prices with vehicle types' multipliers to display the fares for each city pair and vehicle type.
- If the command is to show the available commands and usage information (dm /?):
- Print the list of commands and their usage explanations.

- For other valid commands:

- Extract the start and end cities from the command-line arguments and convert them to uppercase.

- Check if the cities are valid (exist in the price_list dictionary).

- Check for the presence of promo codes and vehicle types in the command-line arguments.

- Calculate the initial price based on the shortest path between the start and end cities.

- Apply promo code reduction (if valid) and calculate the final price.

- Generate random auto-generated reductions for lucky passengers (if applicable).

- Apply the vehicle type price multiplier and update the final price accordingly.

- Create the invoice for the trip using the create_invoice function.

- Print the final trip details, including the fare amount, promo code reduction (if any), random reduction (if any), and the final payment amount.

5. Run the main function with command-line arguments to execute the application.

2.2. Python Code

```
# Importing necessary modules

import random

import datetime

import sys


# Price list dictionary containing the prices between different cities

price_list = {

    'ALVIN': {'ALVIN': 0, 'JAMZ': 20, 'ZUHAR': 20, 'RAZI': 40, 'MALI': 40},

    'JAMZ': {'JAMZ': 0, 'ALVIN': 20, 'RAZI': 20, 'MALI': 40, 'ZUHAR': 40},

    'RAZI': {'RAZI': 0, 'JAMZ': 20, 'MALI': 20, 'ALVIN': 40, 'ZUHAR': 40},
```

```python
    'MALI': {'MALI': 0, 'RAZI': 20, 'ZUHAR': 20, 'ALVIN': 40, 'JAMZ': 40},

    'ZUHAR': {'ZUHAR': 0, 'ALVIN': 20, 'MALI': 20, 'JAMZ': 40, 'RAZI': 40}

}


# Vehicle prices dictionary for different vehicle types

vPrices = {

    'TRISHAW': 1,

    'CAR': 2,

    'VAN': 3}


# Promo codes dictionary with corresponding reduction values

promo = {'pro1': 1,

        'pro2': 2,

        'pro3': 3,

        'pro4': 4,

        'pro5': 5,

        'pro6': 6,

        'pro7': 7,

        'pro8': 8,

        'pro9': 9,

        'pro10': 10,

        'pro11': 11,

        'pro12': 12,
```

```python
    'pro13': 13,

     'pro14': 14,

     'pro15': 15}


# Function to calculate initial price between two cities

def get_initial_price(price_list, start_place, end_place):

    return price_list[start_place][end_place]


# Function to create an invoice file for the trip

def create_invoice(start, end, final_payment, promo_reduction, random_reduction):

    current_datetime = datetime.datetime.now()

    formatted_datetime = current_datetime.strftime("%Y-%m-%d %H_%M_%S_%f")[:24]

    time1 = current_datetime.strftime("%H:%M:%S")[:8]

    filename = f"{formatted_datetime}.txt"

    with open(filename, 'w') as file:

        file.write(f"Date:- {current_datetime.date()}\n")

        file.write(f"Time:- {time1}\n")

        file.write(f"Start:- {start}\n")

        file.write(f"End:- {end}\n")

        file.write(f"Amount:- {price_list[start][end]} KMD\n")

        file.write(f"Promo:- {promo_reduction} KMD\n")

        file.write(f"Random Reduction:- {random_reduction} KMD\n")

        file.write(f"Final payment:- {final_payment} KMD\n")
```

```python
# Main function to handle user commands

def main(args):

    # Checking the minimum requirement for valid commands

    if len(args) < 2:

        print("Invalid command")

        return


    # Command to show the full price plan for all vehicle types

    if args[1] == "/price":

        vehicle_types = ['TRISHAW', 'CAR', 'VAN']

        for vehicle_type in vehicle_types:

            print(f"\nPrice for '{vehicle_type}':")

            for city, prices in price_list.items():

                print(f"Start {city}:")

                for destination, price in prices.items():

                    print(f"End {destination}:- {price * vPrices[vehicle_type]} KMD")

        return


    # Command to show the available commands and usage information

    if args[1] == "/?":

        print("Commands:")
```

```python
        print("dm <Start> <End>:- Shows the price between the two cities and generates an invoice
    file for the trip")

        print()

        print("dm <Start> <End> /pro<code>:- Shows the price between the two cities after
    applying the promo code and generates an invoice file for the trip")

        print()

        print("dm /price:- Show the full price plan for all vehicle types in the whole country")

        print()

        print("dm <Start> <End> /<vehicle>:- Shows the price between the two cities(/c for car , /v
    for a van, Default is Trishaw) and generates an invoice file for the trip")

        print()

        print("dm <Start> <End> /pro<code> /<vehicle>:- Shows the price between the two cities
    after applying the promo code and specifies vehicle type and generates an invoice file for the
    trip")

        print()

        print("dm /? :- Shows the dm commands")

        return


    # Handling other commands with required parameters
    if len(args) < 3:

        print("Invalid command")

        return
```

```python
    # Extracting the start and end places from the command-line arguments and converting them
to uppercase

    start_place = args[1].upper()

    end_place = args[2].upper()


    # Initializing variables for promo code and vehicle type

    promo_code = 0

    vehicle_type = "TRISHAW"


    # Checking if the start and end places are valid cities in the price_list dictionary

    if start_place not in price_list or end_place not in price_list[start_place]:

        print("Invalid command")

        return


    # Check for promo code and vehicle type parameters

    if len(args) > 3 and not start_place == end_place:

        for arg in args[3:]:

            if arg.startswith("/pro"):

                promo_code = arg[1:]

            elif arg.startswith("/"):

                if arg[1:].upper() == 'C':

                    vehicle_type = "CAR"

                elif arg[1:].upper() == "V":
```

```python
            vehicle_type = "VAN"


# Getting the initial price between the start and end places from the price_list dictionary

price = get_initial_price(price_list, start_place, end_place)


# Initializing variables for final calculations

final_price = price

promo_value = 0

random_reduction = 0

luck = 0


 # Apply promo code reduction if valid

if promo_code in promo:

    promo_value = promo[promo_code]

    final_price = price - promo_value

else:

    # Algorithm to generate random reduction

    if not start_place == end_place:

        luck = random.randint(0, 10)

        if luck == 1 or luck == 9:

            random_reduction = 5

            final_price = price - random_reduction
```

```python
    # Apply vehicle type price multiplier and update final price
    if vehicle_type in vPrices:

        price *= vPrices[vehicle_type]

        final_price = price - promo_value - random_reduction


    # Create the invoice for the trip

    create_invoice(start_place, end_place,final_price,promo_value, random_reduction)


    # Print final details

    print(f"Amount:- {price} KMD")

    if promo_code:

        print(f"Promo Value:- {promo_value} KMD")

    if luck == 1 or luck == 9:

        print(f"Random Offer:- {random_reduction} KMD")

    print(f"Final Amount:- {final_price} KMD")


# Check if the script is being run as the main program.

if __name__ == "__main__":

    # Command-line input and calling the main function

    main(sys.argv)
```

2.3. Screenshots of the Python Code


```python
# Importing necessary modules
import random
import datetime
import sys

# Price list dictionary containing the prices between different cities
price_list = {
    'ALVIN': {'ALVIN': 0, 'JAMZ': 20, 'ZUHAR': 20, 'RAZI': 40, 'MALI': 40},
    'JAMZ': {'JAMZ': 0, 'ALVIN': 20, 'RAZI': 20, 'MALI': 40, 'ZUHAR': 40},
    'RAZI': {'RAZI': 0, 'JAMZ': 20, 'MALI': 20, 'ALVIN': 40, 'ZUHAR': 40},
    'MALI': {'MALI': 0, 'RAZI': 20, 'ZUHAR': 20, 'ALVIN': 40, 'JAMZ': 40},
    'ZUHAR': {'ZUHAR': 0, 'ALVIN': 20, 'MALI': 20, 'JAMZ': 40, 'RAZI': 40}
}

# Vehicle prices dictionary for different vehicle types
vPrices = {
    'TRISHAW': 1,
    'CAR': 2,
    'VAN': 3}

# Promo codes dictionary with corresponding reduction values
promo = {'pro1': 1,
         'pro2': 2,
         'pro3': 3,
         'pro4': 4,
         'pro5': 5,
         'pro6': 6,
         'pro7': 7,
         'pro8': 8,
         'pro9': 9,
         'pro10': 10,
         'pro11': 11,
         'pro12': 12,
         'pro13': 13,
         'pro14': 14,
         'pro15': 15}

# Function to calculate initial price between two cities
def get_initial_price(price_list, start_place, end_place):
    return price_list[start_place][end_place]

# Function to create an invoice file for the trip
def create_invoice(start, end, final_payment, promo_reduction, random_reduction):
    current_datetime = datetime.datetime.now()
    formatted_datetime = current_datetime.strftime("%Y-%m-%d %H_%M_%S_%f")[:24]
    time1 = current_datetime.strftime("%H:%M:%S")[:8]
    filename = f"{formatted_datetime}.txt"
```

Figure 1-1st Screenshot of the code


```python
    time1 = current_datetime.strftime("%H:%M:%S")[:8]
    filename = f"{formatted_datetime}.txt"
    with open(filename, 'w') as file:
        file.write(f"Date:- {current_datetime.date()}\n")
        file.write(f"Time:- {time1}\n")
        file.write(f"Start:- {start}\n")
        file.write(f"End:- {end}\n")
        file.write(f"Amount:- {price_list[start][end]} KMD\n")
        file.write(f"Promo:- {promo_reduction} KMD\n")
        file.write(f"Random Reduction:- {random_reduction} KMD\n")
        file.write(f"Final payment:- {final_payment} KMD\n")

# Main function to handle user commands
def main(args):
    # Checking the minimum requirement for valid commands
    if len(args) < 2:
        print("Invalid command")
        return

    # Command to show the full price plan for all vehicle types
    if args[1] == "/price":
        vehicle_types = ['TRISHAW', 'CAR', 'VAN']
        for vehicle_type in vehicle_types:
            print(f"\ndm /pricedm /pricePrice for '{vehicle_type}':")
            for city, prices in price_list.items():
                print(f"Start {city}:")
                for destination, price in prices.items():
                    print(f"End {destination}:- {price * vPrices[vehicle_type]} KMD")
        return

    # Command to show the available commands and usage information
    if args[1] == "/?":
        print("Commands:")
        print("dm <Start> <End>:- Shows the price between the two cities and generates an invoice file for the trip")
        print()
        print("dm <Start> <End> /pro<code>:- Shows the price between the two cities after applying the promo code and generates an invoice file for the trip")
        print()
        print("dm /price:- Show the full price plan for all vehicle types in the whole country")
        print()
        print("dm <Start> <End> /<vehicle>:- Shows the price between the two cities(/c for car , /v for a van, Default is Trishaw) and generates an invoice file for the trip")
        print()
        print("dm <Start> <End> /pro<code> /<vehicle>:- Shows the price between the two cities after applying the promo code and specifies vehicle type and generates an invoice file for")
        print()
        print("dm /? :- Shows the dm commands")
        return

    # Handling other commands with required parameters
```

Figure 2-2nd Screenshot of the code

```
dm.py - C:\Users\nawod\dm.py (3.11.3)
File Edit Format Run Options Window Help
 92     # Handling other commands with required parameters
 93     if len(args) < 3:
 94         print("Invalid command")
 95         return
 96
 97     # Extracting the start and end places from the command-line arguments and converting them to uppercase
 98     start_place = args[1].upper()
 99     end_place = args[2].upper()
100
101     # Initializing variables for promo code and vehicle type
102     promo_code = 0
103     vehicle_type = "TRISHAW"
104
105     # Checking if the start and end places are valid cities in the price_list dictionary
106     if start_place not in price_list or end_place not in price_list[start_place]:
107         print("Invalid command")
108         return
109
110     # Check for promo code and vehicle type parameters
111     if len(args) > 3 and not start_place == end_place:
112         for arg in args[3:]:
113             if arg.startswith("/pro"):
114                 promo_code = arg[1:]
115             elif arg.startswith("/"):
116                 if arg[1:].upper() == 'C':
117                     vehicle_type = "CAR"
118                 elif arg[1:].upper() == 'V':
119                     vehicle_type = "VAN"
120
121     # Getting the initial price between the start and end places from the price_list dictionary
122     price = get_initial_price(price_list, start_place, end_place)
123
124     # Initializing variables for final calculations
125     final_price = price
126     promo_value = 0
127     random_reduction = 0
128     luck = 0
129
130      # Apply promo code reduction if valid
131     if promo_code in promo:
132         promo_value = promo[promo_code]
133         final_price = price - promo_value
134     else:
135         # Algorithm to generate random reduction
136         if not start_place == end_place:
137             luck = random.randint(0, 10)
138             if luck == 1 or luck == 9:
```
Ln: 143 Col: 31

Figure 4-3rd Screenshot of the code


```
138             if luck == 1 or luck == 9:
139                 random_reduction = 5
140                 final_price = price - random_reduction
141
142     # Apply vehicle type price multiplier and update final price
143     if vehicle_type in vPrices:
144         price *= vPrices[vehicle_type]
145         final_price = price - promo_value - random_reduction
146
147     # Create the invoice for the trip
148     create_invoice(start_place, end_place,final_price,promo_value, random_reduction)
149
150     # Print final details
151     print(f"Amount:- {price} KMD")
152     if promo_code:
153         print(f"Promo Value:- {promo_value} KMD")
154     if luck == 1 or luck == 9:
155         print(f"Random Offer:- {random_reduction} KMD")
156     print(f"Final Amount:- {final_price} KMD")
157
158 # Check if the script is being run as the main program.
159 if __name__ == "__main__":
160     # Command-line input and calling the main function
161     main(sys.argv)
162
```
Ln: 143 Col: 31

Figure 3-4th Screenshot of the code

2.4.   Python Modules which used to create code

'random' module:

The 'random' module provides various functions to generate random numbers and make decisions based on randomness. This module is used in the code to create a random reduction in the final price of the trip. The 'random.randint(a, b)' function is used to generate random integers

12

between 'a' and 'b' (inclusive). The code uses it to simulate a random event with a 1 in 5 (10%) chance of offering a discount of 5 KMD on the trip.

'datetime' module:

The 'datetime' module in Python provides classes for manipulating dates and times. In the code, this module is used to get the current date and time to include in the invoice file. Specifically, 'datetime.datetime.now()' is used to get the current date and time, and 'strftime()' is used to format it according to the desired format. The current date and time are then added to the invoice file when it is created.
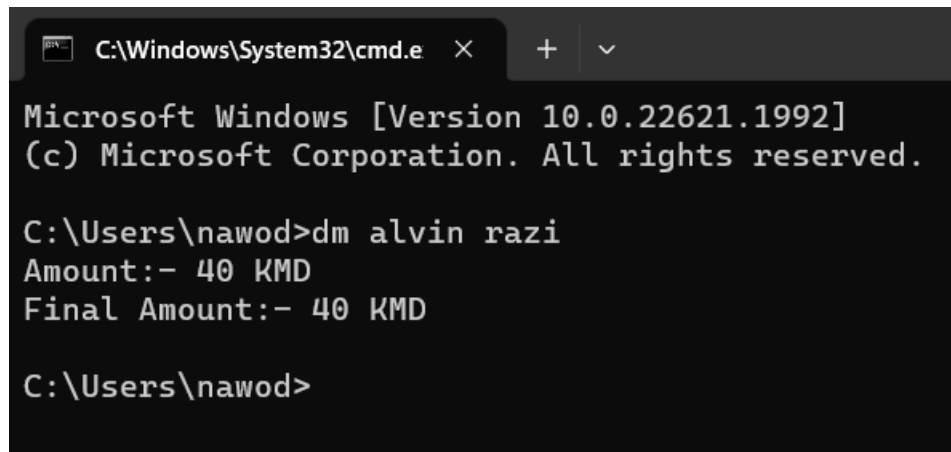
'sys' module:

The 'sys' module in Python provides access to some variables used or maintained by the interpreter and to functions that interact with the Python runtime environment. In the code, 'sys.argv' is used to get the command-line arguments passed when running the script. This allows the script to accept user commands and arguments directly from the command line. These arguments are used to determine the start and end places, promo codes, and vehicle types for the trip.

In summary, the code uses the 'random' module to generate random reductions, the 'datetime' module to include the current date and time in the invoice file, and the 'sys' module to accept user commands and arguments from the command line for processing and generating the trip invoice accordingly.

3. Screenshots of the program output

1. The first command type shows the price between the two cities and generates an invoice file for the trip.
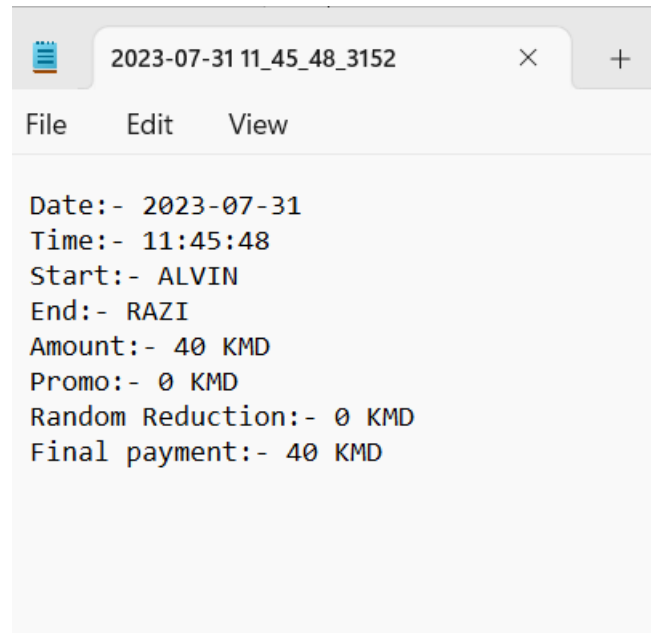
Figure 5-cmd output for first command

Figure 6-Invoice file for first command

2. The second command type shows the price between the two cities after applying the promo code. The promo code amount is deducted from the final bill value. This will generate an invoice file for the trip with promo details.

```
C:\Users\nawod>dm mali zuhar /pro8
Amount:- 20 KMD
Promo Value:- 8 KMD
Final Amount:- 12 KMD

C:\Users\nawod>
```

Figure 7-cmd output for second command

```
2023-07-31 11_56_46_4879              ×     +

File    Edit    View

Date:- 2023-07-31
Time:- 11:56:46
Start:- MALI
End:- ZUHAR
Amount:- 20 KMD
Promo:- 8 KMD
Random Reduction:- 0 KMD
Final payment:- 12 KMD
```
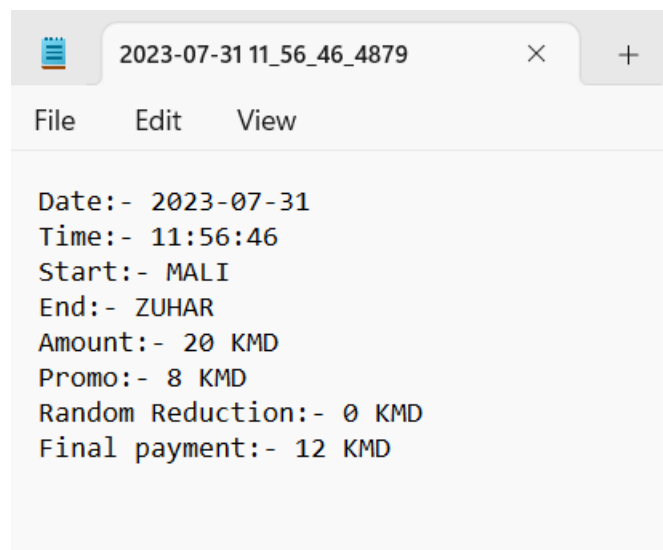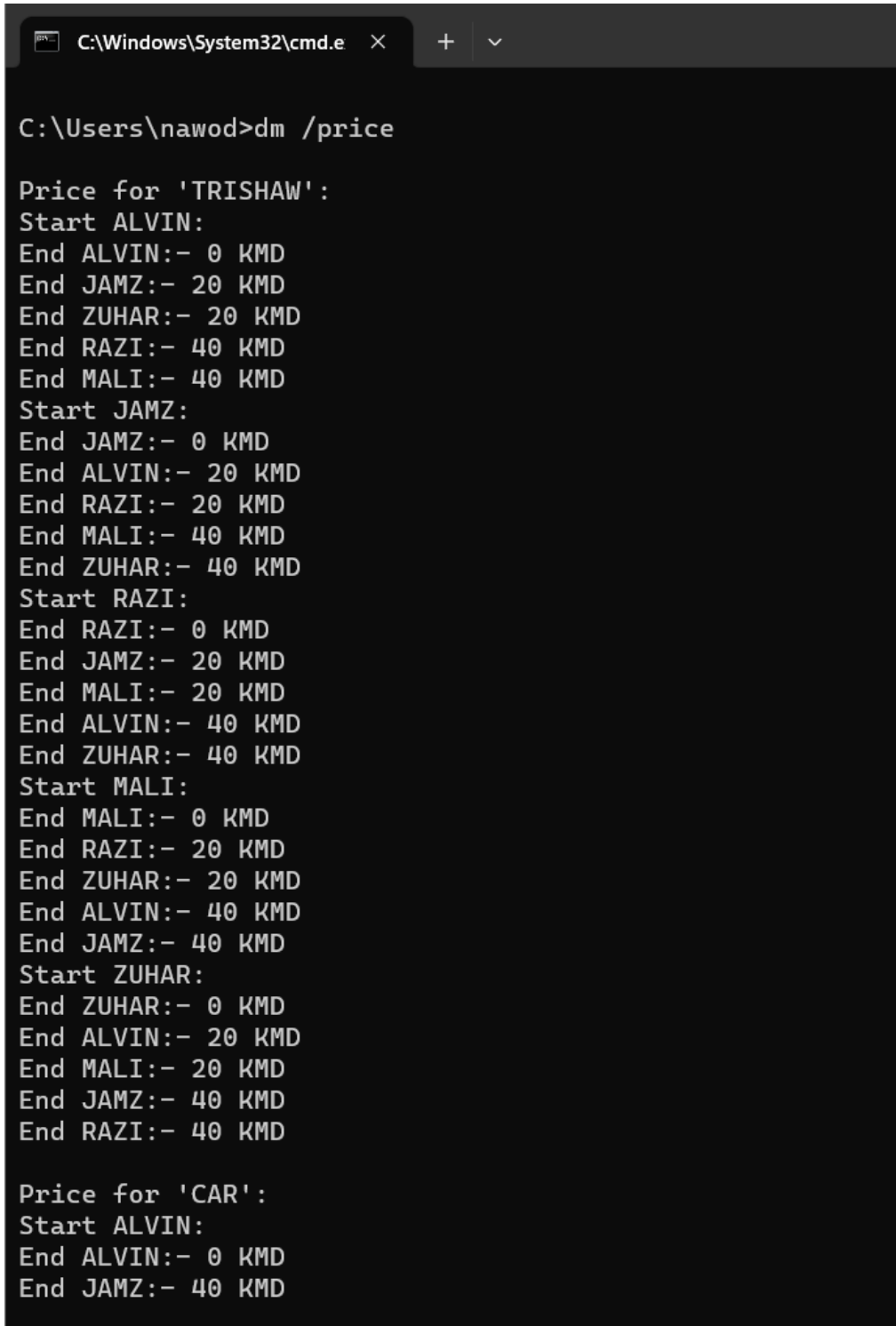
Figure 8-invoice file for second command

3. The third command type show the full price plan for the whole country. (For all 3 vehicles)
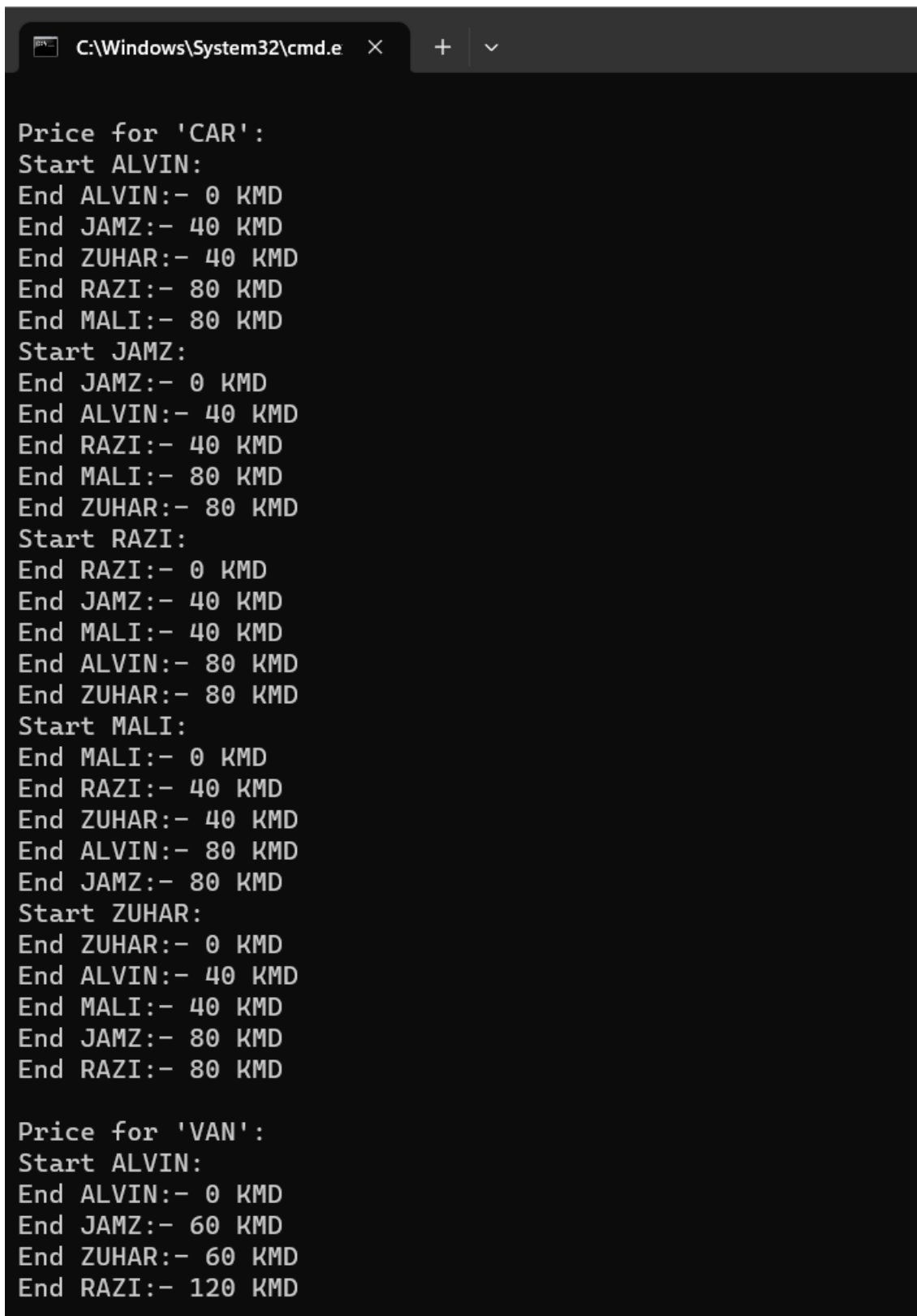
```
C:\Windows\System32\cmd.e    ×    +    ∨

C:\Users\nawod>dm /price

Price for 'TRISHAW':
Start ALVIN:
End ALVIN:- 0 KMD
End JAMZ:- 20 KMD
End ZUHAR:- 20 KMD
End RAZI:- 40 KMD
End MALI:- 40 KMD
Start JAMZ:
End JAMZ:- 0 KMD
End ALVIN:- 20 KMD
End RAZI:- 20 KMD
End MALI:- 40 KMD
End ZUHAR:- 40 KMD
Start RAZI:
End RAZI:- 0 KMD
End JAMZ:- 20 KMD
End MALI:- 20 KMD
End ALVIN:- 40 KMD
End ZUHAR:- 40 KMD
Start MALI:
End MALI:- 0 KMD
End RAZI:- 20 KMD
End ZUHAR:- 20 KMD
End ALVIN:- 40 KMD
End JAMZ:- 40 KMD
Start ZUHAR:
End ZUHAR:- 0 KMD
End ALVIN:- 20 KMD
End MALI:- 20 KMD
End JAMZ:- 40 KMD
End RAZI:- 40 KMD

Price for 'CAR':
Start ALVIN:
End ALVIN:- 0 KMD
End JAMZ:- 40 KMD
```

Figure 9-cmd output 1 for third command
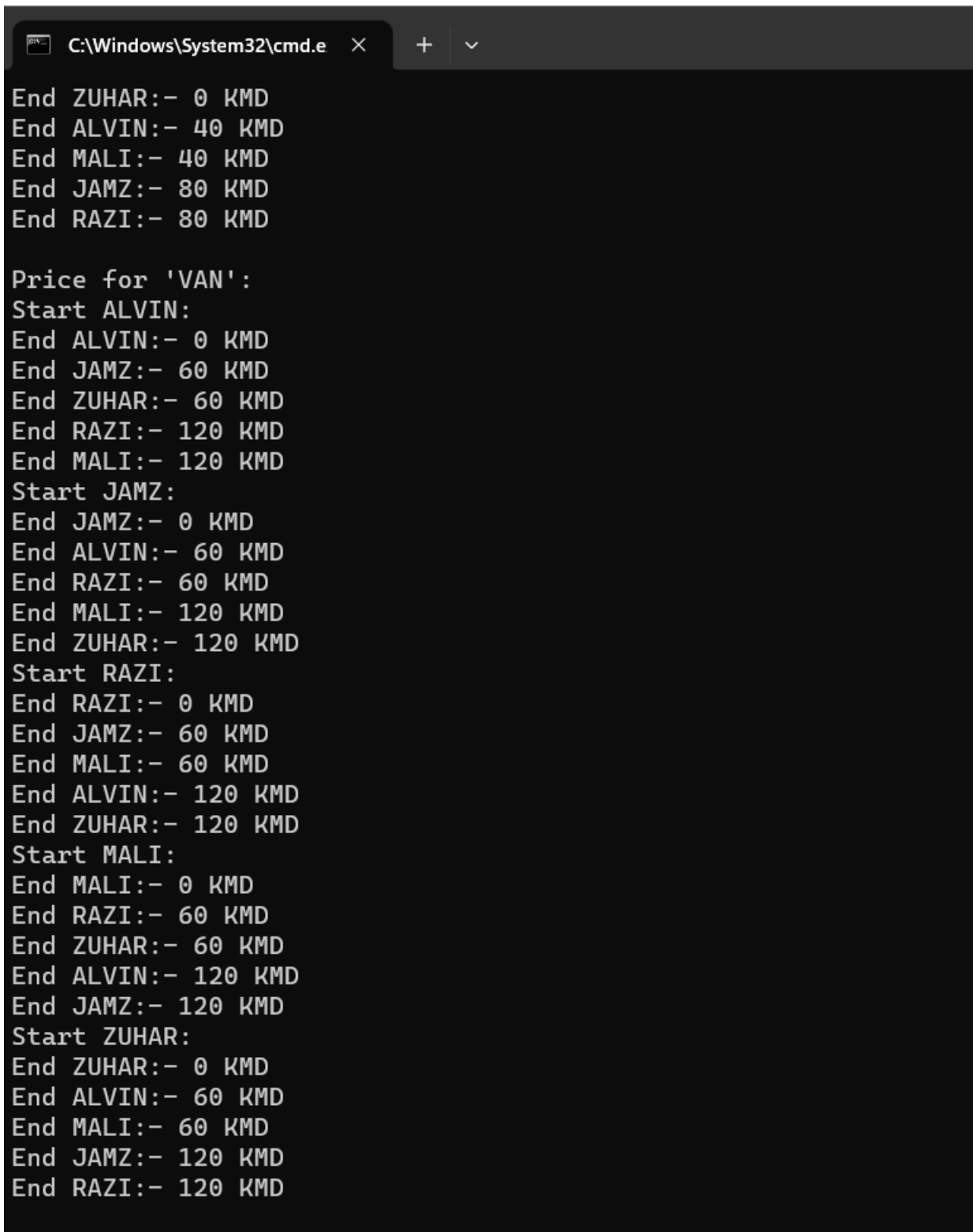
```
Price for 'CAR':
Start ALVIN:
End ALVIN:- 0 KMD
End JAMZ:- 40 KMD
End ZUHAR:- 40 KMD
End RAZI:- 80 KMD
End MALI:- 80 KMD
Start JAMZ:
End JAMZ:- 0 KMD
End ALVIN:- 40 KMD
End RAZI:- 40 KMD
End MALI:- 80 KMD
End ZUHAR:- 80 KMD
Start RAZI:
End RAZI:- 0 KMD
End JAMZ:- 40 KMD
End MALI:- 40 KMD
End ALVIN:- 80 KMD
End ZUHAR:- 80 KMD
Start MALI:
End MALI:- 0 KMD
End RAZI:- 40 KMD
End ZUHAR:- 40 KMD
End ALVIN:- 80 KMD
End JAMZ:- 80 KMD
Start ZUHAR:
End ZUHAR:- 0 KMD
End ALVIN:- 40 KMD
End MALI:- 40 KMD
End JAMZ:- 80 KMD
End RAZI:- 80 KMD

Price for 'VAN':
Start ALVIN:
End ALVIN:- 0 KMD
End JAMZ:- 60 KMD
End ZUHAR:- 60 KMD
End RAZI:- 120 KMD
```

Figure 10-cmd output 2 for third command

```
End ZUHAR:- 0 KMD
End ALVIN:- 40 KMD
End MALI:- 40 KMD
End JAMZ:- 80 KMD
End RAZI:- 80 KMD

Price for 'VAN':
Start ALVIN:
End ALVIN:- 0 KMD
End JAMZ:- 60 KMD
End ZUHAR:- 60 KMD
End RAZI:- 120 KMD
End MALI:- 120 KMD
Start JAMZ:
End JAMZ:- 0 KMD
End ALVIN:- 60 KMD
End RAZI:- 60 KMD
End MALI:- 120 KMD
End ZUHAR:- 120 KMD
Start RAZI:
End RAZI:- 0 KMD
End JAMZ:- 60 KMD
End MALI:- 60 KMD
End ALVIN:- 120 KMD
End ZUHAR:- 120 KMD
Start MALI:
End MALI:- 0 KMD
End RAZI:- 60 KMD
End ZUHAR:- 60 KMD
End ALVIN:- 120 KMD
End JAMZ:- 120 KMD
Start ZUHAR:
End ZUHAR:- 0 KMD
End ALVIN:- 60 KMD
End MALI:- 60 KMD
End JAMZ:- 120 KMD
End RAZI:- 120 KMD
```

Figure 12-cmd output 3 for third command

4. The fourth command type shows the price between the two cities and generates an invoice file for the trip. This uses a car (/c for car while /v for a van. Default is Trishaw)

```
C:\Users\nawod>dm razi mali /c
Amount:- 40 KMD
Random Offer:- 5 KMD
Final Amount:- 35 KMD

C:\Users\nawod>dm razi mali /c
Amount:- 40 KMD
Final Amount:- 40 KMD

C:\Users\nawod>dm razi mali /v
Amount:- 60 KMD
Final Amount:- 60 KMD

C:\Users\nawod>
```

Figure 13-cmd output for fourth command

```
2023-07-31 12_28_40_2283                    ×

File    Edit    View

Date:- 2023-07-31
Time:- 12:28:40
Start:- RAZI
End:- MALI
Amount:- 20 KMD
Promo:- 0 KMD
Random Reduction:- 5 KMD
Final payment:- 35 KMD
```
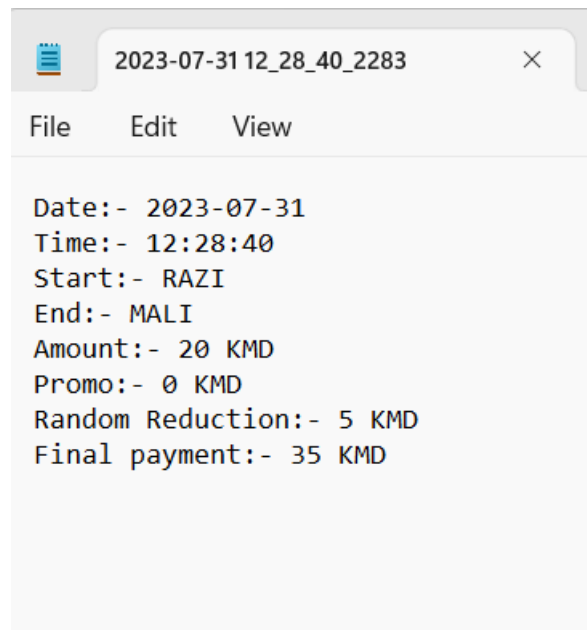
Figure 14-Invoice file for fourth command with random reduction
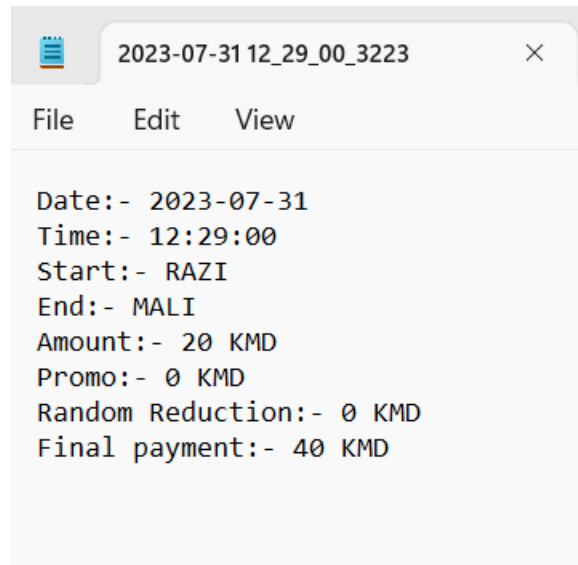
Figure 15-Invoice file for fourth command with car vehicle type
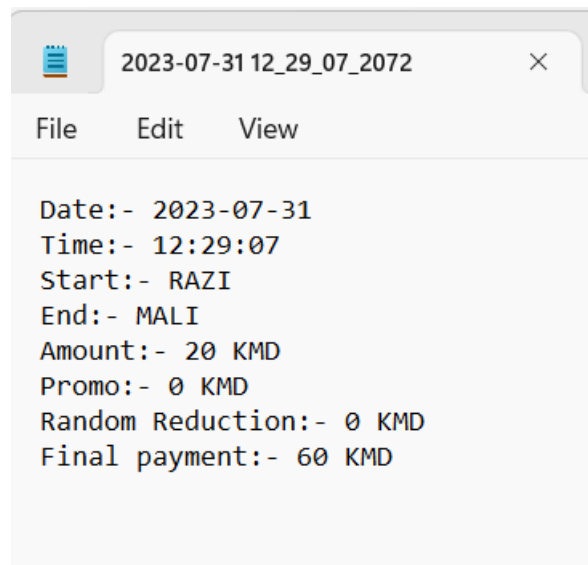
Figure 16-Invoice file for fourth command with van vehicle type

5. The Fifth command type shows the price between the two cities while user apply a promo reduction and the prefer vehicle s and generates an invoice file for the trip .

```
C:\Users\nawod>dm alvin razi /pro10 /v
Amount:- 120 KMD
Promo Value:- 10 KMD
Final Amount:- 110 KMD

C:\Users\nawod>dm alvin razi /pro10 /c
Amount:- 80 KMD
Promo Value:- 10 KMD
Final Amount:- 70 KMD

C:\Users\nawod>
```

Figure 17-cmd output for fifth command

```
2023-07-31 12_48_34_8552        ×

File    Edit    View

Date:- 2023-07-31
Time:- 12:48:34
Start:- ALVIN
End:- RAZI
Amount:- 40 KMD
Promo:- 10 KMD
Random Reduction:- 0 KMD
Final payment:- 110 KMD
```
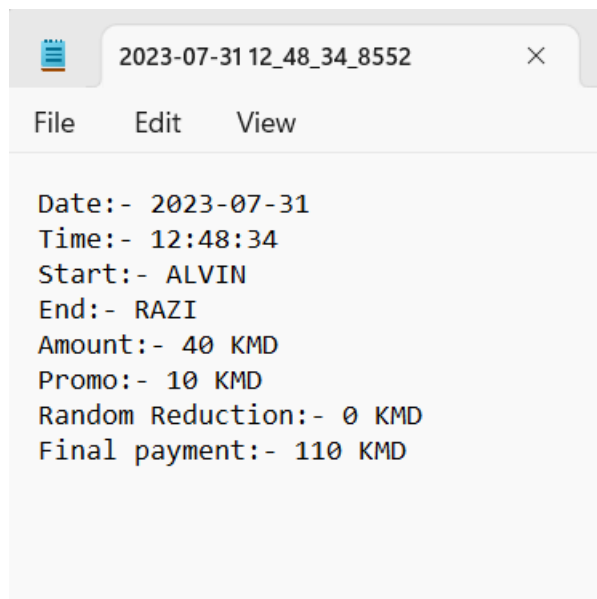
Figure 19-Invoice file for fifth command with van

```
2023-07-31 12_48_42_8950        ×

File    Edit    View

Date:- 2023-07-31
Time:- 12:48:42
Start:- ALVIN
End:- RAZI
Amount:- 40 KMD
Promo:- 10 KMD
Random Reduction:- 0 KMD
Final payment:- 70 KMD
```
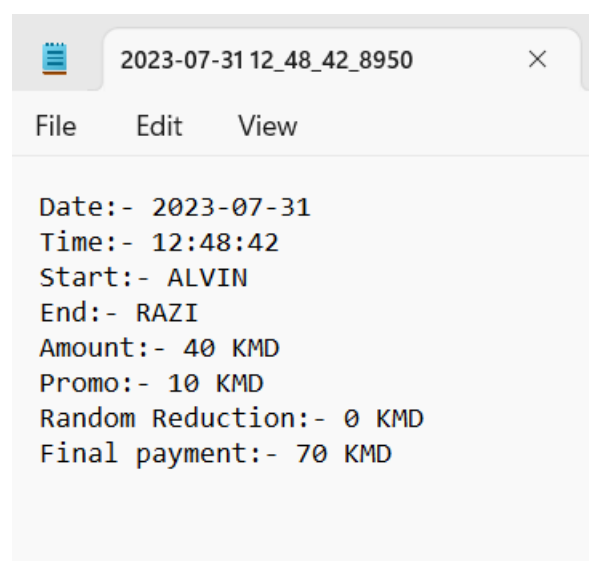
Figure 18-Invoice file for fifth command with car

6. The sixth command type shows how this "dm" command functions.

```
C:\Users\nawod>dm /?
Commands:
dm <Start> <End>:- Shows the price between the two cities and generates an i
nvoice file for the trip

dm <Start> <End> /pro<code>:- Shows the price between the two cities after a
pplying the promo code and generates an invoice file for the trip

dm /price:- Show the full price plan for all vehicle types in the whole coun
try

dm <Start> <End> /<vehicle>:- Shows the price between the two cities(/c for
car , /v for a van, Default is Trishaw) and generates an invoice file for th
e trip

dm <Start> <End> /pro<code> /<vehicle>:- Shows the price between the two cit
ies after applying the promo code and specifies vehicle type and generates a
n invoice file for the trip

dm /? :- Shows the dm commands

C:\Users\nawod>
```

Figure 20-cmd output for sixth command

4. Test cases used to test the program

Test Case ID	Test Scenario	Input command	Expected Results	Actual Results	Pass/Fail
01	Shows the price between the alvin and razi	dm alvin razi	Show the Amount, Final Amount and generates an invoice file for the trip	As Expected, but generated the Random offer	Pass
02	Shows the price between the jamz and mali	dm jamz mali	Show the Amount, Final Amount and generates an invoice file for the trip	As Expected	Pass
03	Shows the price within the zuhar	dm zuhar zuhar	Show the Amount = 0, Final Amount = 0 and generates an invoice file for the trip	As Expected	Pass
04	Shows the price between the alvin and razi after applying the promo code 2	dm alvin razi /pro2	Show the Amount, Promo Value and Final Amount and generates an invoice file for the trip	As Expected	Pass
05	Show the full price plan for the whole country (for all 3 vehicles)	dm /price	Show the full price plan	As Expected	Pass
06	Shows the price between the jamz and	dm jamz zuhar /c	Show the amount and final amount	As Expected	Pass

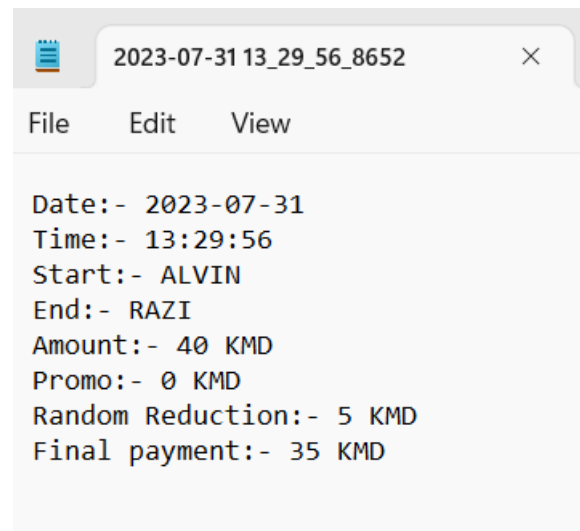|	zuhar with the car vehicle type command.		and generates an invoice file for the trip		
07	Shows the price between the alvin and razi while applying a 10 KMD reduction to total bill. The rider prefers a van.	dm alvin razi /pro10 /v	Show the amount, promo value and final amount and generates an invoice file for the trip	As Expected	Pass
08	Show how this "dm" command functions	dm /?	Show the command and description	As Expected	Pass
09	Enter a wrong command	dm /alvin /?	Invalid command	As Expected	Pass

Table 1-Test Cases

```
C:\Users\nawod>dm alvin razi
Amount:- 40 KMD
Random Offer:- 5 KMD
Final Amount:- 35 KMD

C:\Users\nawod>
```

Figure 22-Test case 1 cmd

```
2023-07-31 13_29_56_8652         ×

File    Edit    View

Date:- 2023-07-31
Time:- 13:29:56
Start:- ALVIN
End:- RAZI
Amount:- 40 KMD
Promo:- 0 KMD
Random Reduction:- 5 KMD
Final payment:- 35 KMD
```

Figure 21-Test case 1 invoice file

```
C:\Users\nawod>dm jamz mali
Amount:- 40 KMD
Final Amount:- 40 KMD

C:\Users\nawod>
```

Figure 23-Test case 2 cmd

File   Edit   View

```
Date:- 2023-07-31
Time:- 13:35:37
Start:- JAMZ
End:- MALI
Amount:- 40 KMD
Promo:- 0 KMD
Random Reduction:- 0 KMD
Final payment:- 40 KMD
```
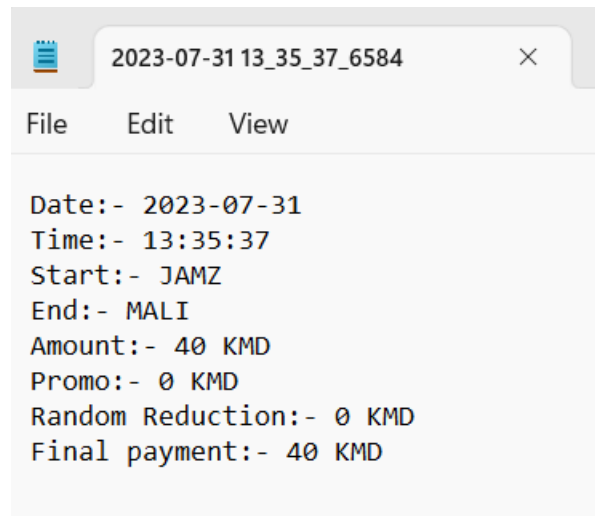
Figure 24- Test case 2 invoice file

```
C:\Users\nawod>dm zuhar zuhar
Amount:- 0 KMD
Final Amount:- 0 KMD

C:\Users\nawod>
```

Figure 26-test case 3 cmd

File   Edit   View

```
Date:- 2023-07-31
Time:- 13:47:15
Start:- ZUHAR
End:- ZUHAR
Amount:- 0 KMD
Promo:- 0 KMD
Random Reduction:- 0 KMD
Final payment:- 0 KMD
```
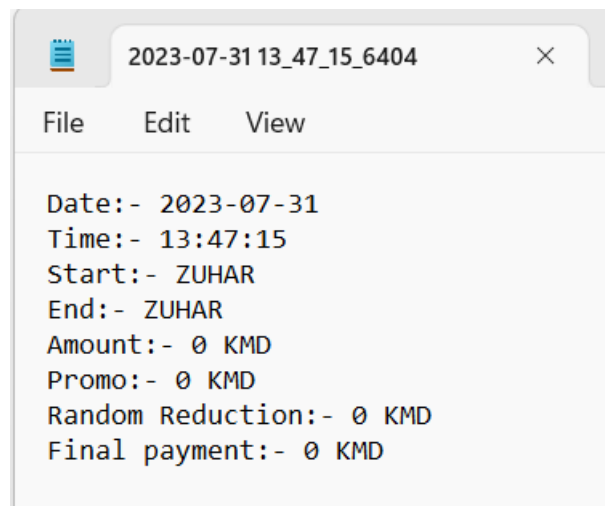
Figure 25-Test case 3 invoice file

```
C:\Users\nawod>dm alvin razi /pro2
Amount:- 40 KMD
Promo Value:- 2 KMD
Final Amount:- 38 KMD


C:\Users\nawod>
```

Figure 28-Test case 4 cmd


```
2023-07-31 13_54_44_9937                    ×

File    Edit    View

Date:- 2023-07-31
Time:- 13:54:44
Start:- ALVIN
End:- RAZI
Amount:- 40 KMD
Promo:- 2 KMD
Random Reduction:- 0 KMD
Final payment:- 38 KMD
```

Figure 27-Test case 4 Invoice file


```
C:\Windows\System32\cmd.e   ×

C:\Users\nawod>dm /price

Price for 'TRISHAW':
Start ALVIN:
End ALVIN:- 0 KMD
End JAMZ:- 20 KMD
End ZUHAR:- 20 KMD
End RAZI:- 40 KMD
End MALI:- 40 KMD
Start JAMZ:
End JAMZ:- 0 KMD
End ALVIN:- 20 KMD
End RAZI:- 20 KMD
End MALI:- 40 KMD
End ZUHAR:- 40 KMD
Start RAZI:
End RAZI:- 0 KMD
End JAMZ:- 20 KMD
End MALI:- 20 KMD
End ALVIN:- 40 KMD
End ZUHAR:- 40 KMD
Start MALI:
End MALI:- 0 KMD
End RAZI:- 20 KMD
End ZUHAR:- 20 KMD
End ALVIN:- 40 KMD
End JAMZ:- 40 KMD
Start ZUHAR:
End ZUHAR:- 0 KMD
End ALVIN:- 20 KMD
End MALI:- 20 KMD
End JAMZ:- 40 KMD
```

Figure 30-Test case 5 cmd for Trishaws


```
C:\Windows\System32\cmd.e   ×

Price for 'CAR':
Start ALVIN:
End ALVIN:- 0 KMD
End JAMZ:- 40 KMD
End ZUHAR:- 40 KMD
End RAZI:- 80 KMD
End MALI:- 80 KMD
Start JAMZ:
End JAMZ:- 0 KMD
End ALVIN:- 40 KMD
End RAZI:- 40 KMD
End MALI:- 80 KMD
End ZUHAR:- 80 KMD
Start RAZI:
End RAZI:- 0 KMD
End JAMZ:- 40 KMD
End MALI:- 40 KMD
End ALVIN:- 80 KMD
End ZUHAR:- 80 KMD
Start MALI:
End MALI:- 0 KMD
End RAZI:- 40 KMD
End ZUHAR:- 40 KMD
End ALVIN:- 80 KMD
End JAMZ:- 80 KMD
Start ZUHAR:
End ZUHAR:- 0 KMD
End ALVIN:- 40 KMD
End MALI:- 40 KMD
End JAMZ:- 80 KMD
```

Figure 31-Test case 5 cmd for cars


```
C:\Windows\System32\cmd.e   ×

Price for 'VAN':
Start ALVIN:
End ALVIN:- 0 KMD
End JAMZ:- 60 KMD
End ZUHAR:- 60 KMD
End RAZI:- 120 KMD
End MALI:- 120 KMD
Start JAMZ:
End JAMZ:- 0 KMD
End ALVIN:- 60 KMD
End RAZI:- 60 KMD
End MALI:- 120 KMD
End ZUHAR:- 120 KMD
Start RAZI:
End RAZI:- 0 KMD
End JAMZ:- 60 KMD
End MALI:- 60 KMD
End ALVIN:- 120 KMD
End ZUHAR:- 120 KMD
Start MALI:
End MALI:- 0 KMD
End RAZI:- 60 KMD
End ZUHAR:- 60 KMD
End ALVIN:- 120 KMD
End JAMZ:- 120 KMD
Start ZUHAR:
End ZUHAR:- 0 KMD
End ALVIN:- 60 KMD
End MALI:- 60 KMD
End JAMZ:- 120 KMD
```
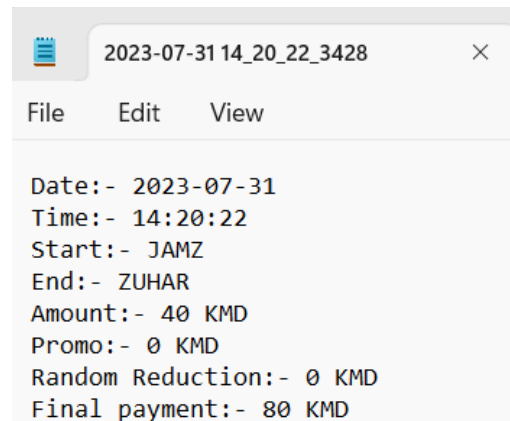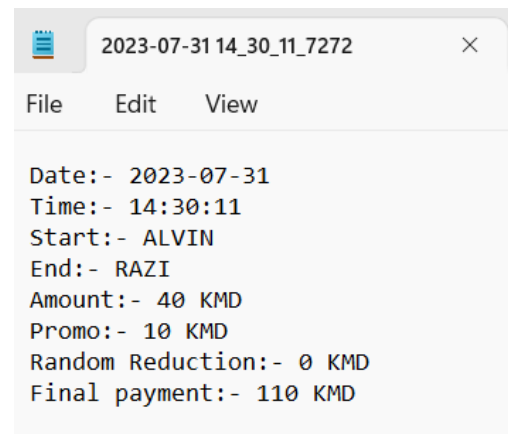
Figure 29-test case 5 cmd for van

Figure 33-Test case 6 cmd for

Figure 32-Test case 6 invoice file

Figure 35-Test case 7 cmd

Figure 34-Test case 7 invoice file

Figure 36- test case 8 cmd

Figure 37- Test case 9 cmd

5. Conclusion

Python code is a console application for a fictional country called "The Kingdom of Miranda." The application allows users to view trip details and generate invoices for cab services provided by the DropMe company. The country has five main cities: Alvin (Capital), Jamz, Razi, Mali, and Zuhar. The official currency is KMD (Kingdom of Miranda Dollars).

The application supports various commands from the client side, such as:

- Displaying the price between two cities.
- Applying promo codes to get discounts on the trip.
- Choosing different vehicle types (Trishaw, Car, Van) for the cab service.
- Showing the full price plan for all vehicle types in the whole country.
- Displaying usage information for the commands.

The price for traveling between cities is pre-defined in a price chart, and the application calculates the cost based on the selected vehicle type and any applicable promo codes or random offers. It also generates an invoice for each trip, recording details like date, time, start, end, amount, promo value, random reduction, and final payment.

The application supports promo codes with different reduction values ranging from 1 KMD to 15 KMD. Additionally, the system randomly provides special promotions at a fixed price of 5 KMD for lucky passengers, which is not applicable if the passenger uses a promotional code.

Overall, the Python code is well-structured and efficiently handles different scenarios to provide accurate trip details and generate invoices for the passengers of The Kingdom of Miranda.