Data Science Assignment 1

Muhammad Abdullah (22P-9371)

February 2025

1 Problem 1: Find Two Numbers Adding Up to a Target Sum

1.1 Problem Statement

The task is to write a Python program that takes a list of numbers from the user and finds two numbers in the list that add up to a target sum, also provided by the user. The program should then return the positions of the two numbers in the list and the numbers themselves.

1.2 Approach

To solve this problem, I wrote a program that looks at every pair of numbers in the list. It checks if the sum of each pair equals the target sum. If such a pair is found, the program will return the positions (indexes) of the numbers and the numbers themselves. If no pair adds up to the target sum, the program will inform the user that no pair was found.

1.3 Code

```
def checkSum(l1, target):
    for i in range(len(l1)):
        for j in range(i+1, len(l1)):
            if l1[i] + l1[j] == target:
                return i, j, l1[i], l1[j]

_len = int(input("Enter the number of elements: "))
listofIntegers = []
for i in range(_len):
    num = int(input(f"Enter number {i+1}: "))
    listofIntegers.append(num)
targetSum = int(input("Enter the target sum: "))
check = checkSum(listofIntegers, targetSum)

if check:
    index1, index2, element1, element2 = check
    print(f"First index is {index1}\nSecond index is {index2}\nTwo numbers that add up the
    targeted Sum are {element1} and {element2}.")
else:
    print("No pair found")
```

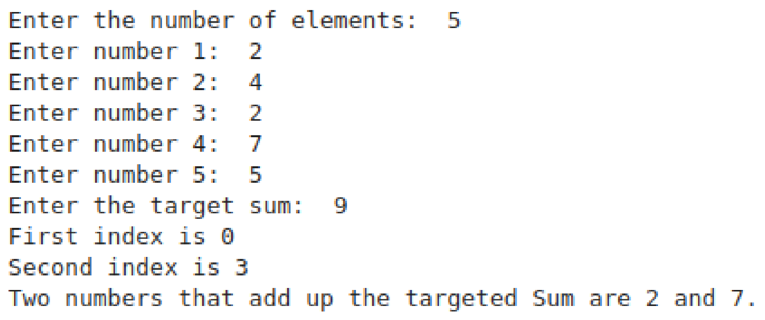
1.4 Explanation

The program first defines a function `checkSum(l1, target)` that goes through every possible pair of numbers in the list. It uses two loops: the outer loop starts with the first number and the inner loop checks every number that comes after it. For each pair of numbers, the program checks if their sum equals the target sum. If it finds such a pair, it returns the indices (positions) of these two numbers and their values. The program then asks the user to enter the list of numbers and the target sum. It calls the `checkSum` function to find a pair that adds up to the target.

If a pair is found, it displays the indices and values of the two numbers. If no pair is found, it shows a message saying that no such pair exists.

1.5 Output

Here is the screenshot of the program's output:



```
Enter the number of elements:  5
Enter number 1:  2
Enter number 2:  4
Enter number 3:  2
Enter number 4:  7
Enter number 5:  5
Enter the target sum:  9
First index is 0
Second index is 3
Two numbers that add up the targeted Sum are 2 and 7.
```

Figure 1: Screenshot of the output

1.6 Conclusion

The program correctly finds two numbers in the list that add up to the target sum. It displays both the positions and the values of the numbers. This method is straightforward and easy to understand, but it could be less efficient if the list contains many numbers because it checks each possible pair. A more advanced approach could improve performance for larger lists.

2 Problem 2: Find Unique Words Sorted by Frequency

2.1 Problem Statement

The task is to write a Python program that takes a list of words as input and returns a list of the unique words in descending order of their frequency in the input list. The program should also print the frequency of each unique word.

2.2 Approach

To solve this, I wrote a program that:
1. Accepts a list of words from the user.
2. Counts the frequency of each word in the list.
3. Sorts the unique words based on their frequency in descending order.
4. Displays each word with its frequency.

2.3 Code

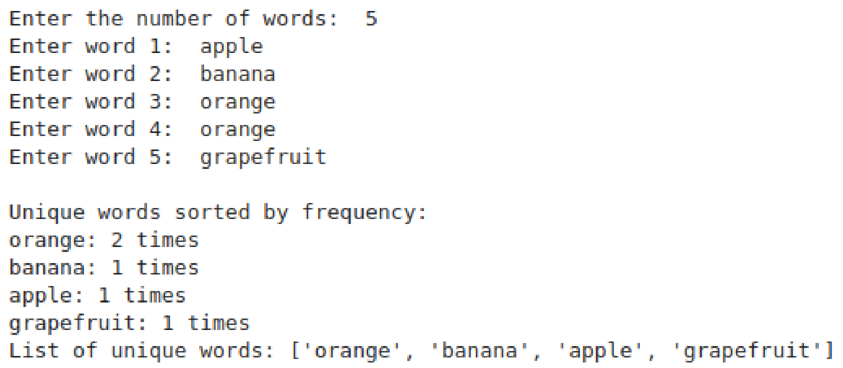
```python
def word_frequency():
    _len = int(input("Enter the number of words: "))
    words = []
    for i in range(_len):
        word = input(f"Enter word {i+1}: ")
        words.append(word)

    uniqueWords = []
    frequencies = []

    for word in words:
        if word in uniqueWords:
            index = uniqueWords.index(word)
            frequencies[index] += 1
        else:
            uniqueWords.append(word)
            frequencies.append(1)

    # Sorting words by frequency in descending order
    for i in range(len(frequencies)):
        max_index = i
        for j in range(i + 1, len(frequencies)):
            if frequencies[j] > frequencies[max_index]:
                max_index = j

        frequencies[i], frequencies[max_index] = frequencies[max_index], frequencies[i]
        uniqueWords[i], uniqueWords[max_index] = uniqueWords[max_index], uniqueWords[i]

    return uniqueWords, frequencies

uniqueWords, frequencies = word_frequency()
print("\nUnique words sorted by frequency:")
for i in range(len(uniqueWords)):
    word = uniqueWords[i]
    freq = frequencies[i]
    print(f"{word}: {freq} times")
print("List of unique words:", uniqueWords)
```

2.4 Explanation

In this program:
- We first take the list of words from the user.
- We then count how often each word appears in the list.
- The program keeps track of each unique word in the `uniqueWords` list and its frequency in the `frequencies` list.
- After counting the frequencies, the program sorts the words based on their frequency in descending order.
- Finally, the program prints out the unique words along with their frequencies.

2.5 Output

Here is the screenshot of the program's output:



```
Enter the number of words:  5
Enter word 1:  apple
Enter word 2:  banana
Enter word 3:  orange
Enter word 4:  orange
Enter word 5:  grapefruit

Unique words sorted by frequency:
orange: 2 times
banana: 1 times
apple: 1 times
grapefruit: 1 times
List of unique words: ['orange', 'banana', 'apple', 'grapefruit']
```

Figure 2: Screenshot of the output

2.6 Conclusion

This program efficiently counts the frequency of each word in the list and sorts them by their frequency. It can be useful for analyzing word frequency in any given input, such as text analysis or data cleaning tasks.

3 Problem 3: Find the Longest Consecutive Sequence

3.1 Problem Statement

The task is to write a Python program that takes a list of integers as input and returns the longest consecutive sequence of numbers in the list.

3.2 Approach

To solve this problem, I wrote a program that:
1. Takes a list of integers as input from the user.
2. Iterates through the list, checking for consecutive numbers.
3. Tracks the longest consecutive sequence encountered during the iteration.
4. Returns the longest consecutive sequence at the end.

The approach uses a loop to compare each number with the next one. If the next number is one greater than the current number, the sequence continues. If not, the sequence ends and is compared with the longest sequence found so far.

3.3 Code

```python
def longest_consecutive(nums):
    if not nums:
        return []

    longest_consecutive_sequence = []
    current_consecutive_sequence = []

    for i in range(len(nums)):
        if not current_consecutive_sequence:
            current_consecutive_sequence.append(nums[i])
        if i + 1 < len(nums) and nums[i + 1] == nums[i] + 1:
            current_consecutive_sequence.append(nums[i + 1])
        else:
            if len(current_consecutive_sequence) > len(longest_consecutive_sequence):
                longest_consecutive_sequence = current_consecutive_sequence

            current_consecutive_sequence = []
    return longest_consecutive_sequence

_len = int(input("Enter the number of integers you want to input: "))
l1 = []
for i in range(_len):
    num = int(input(f"Enter integer {i+1}: "))
    l1.append(num)
sequence = longest_consecutive(l1)
print("Longest consecutive sequence:", sequence)
```

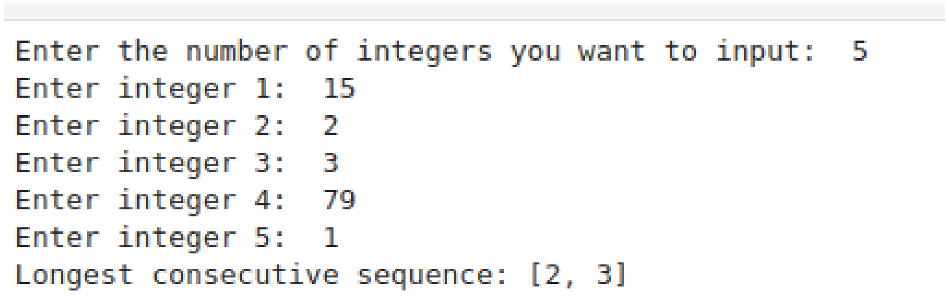
3.4 Explanation

The function `longest_consecutive(nums)` aims to find the longest consecutive sequence of numbers in the list. It uses two lists:
- `current_consecutive_sequence` to track the sequence as we iterate through the list.

- `longest_consecutive_sequence` to store the longest sequence found.
As we loop through the list:
1. If the current number is part of a consecutive sequence, it is added to `current_consecutive_sequence`.
2. If the next number is not consecutive (i.e., it is not one more than the current number), we compare the length of `current_consecutive_sequence` with the longest found so far, and update the result if necessary.
At the end of the loop, the function returns the longest consecutive sequence.

3.5 Output

Here is the screenshot of the program's output:



```
Enter the number of integers you want to input:  5
Enter integer 1:  15
Enter integer 2:  2
Enter integer 3:  3
Enter integer 4:  79
Enter integer 5:  1
Longest consecutive sequence: [2, 3]
```

Figure 3: Screenshot of the output

3.6 Conclusion

This program successfully identifies the longest consecutive sequence of integers from a list. The algorithm is simple but effective, making it easy to understand and use for such tasks. It can be extended to handle other types of sequences as well.

4 Problem 4: Frequency Distribution of Emails Based on Hour

4.1 Problem Statement

The task is to write a Python program that processes a file called `Data.txt` to determine the frequency distribution of emails based on the hour of the day. The program extracts the hour from lines that begin with "From" by locating the timestamp and splitting it using a colon to isolate the hour.

4.2 Approach

To solve this problem, the program performs the following steps:
1. It opens the file `Data.txt` and processes each line.
2. For lines starting with "From ", the program splits the line into words and extracts the timestamp (located at the 5th position).
3. The timestamp is split by the colon, and the hour is isolated.
4. A dictionary `hour_counts` is used to store the count of emails for each hour.
5. Finally, the program prints the frequency of emails for each hour in ascending order.

4.3 Code

```
def ExtractingHours(datafile):
    hour_count = {}

    with open(datafile, "r") as file:
        for line in file:
            if line.startswith("From "):
                words = line.split()
                timestamp = words[5]
                hour = timestamp.split(":")[0]
                hour_count[hour] = hour_count.get(hour, 0) + 1
    for hour, count in sorted(hour_count.items()):
        print(hour, count)

DataFile = "Data.txt"
ExtractingHours(DataFile)
```

4.4 Explanation

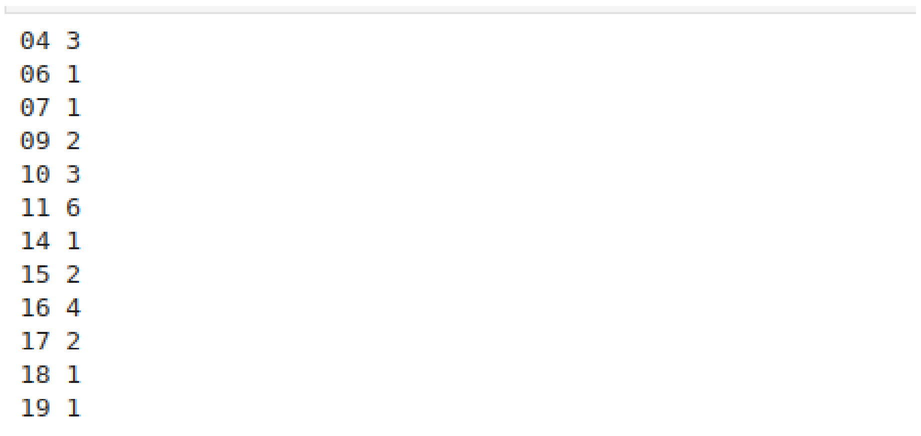
The function `process_email_hours(datafile)` reads the content of the file and processes it line by line:
- If a line starts with "From " (indicating it's an email), the program extracts the timestamp (which is the 5th element in the line when split by spaces).
- The timestamp is in the format `HH:MM:SS`, and by splitting it at the colon, we can isolate the hour (the first part before the colon).
- The program then uses a dictionary, `hour_counts`, to store the frequency of each hour.
- Finally, the dictionary is sorted by the hour, and the frequency distribution is printed.

4.5 Conclusion

This program efficiently processes the email data and calculates the frequency distribution of emails based on the hour. It is useful for understanding email patterns over the course of the day. The program can be extended to handle additional analysis, such as determining the busiest times for emails or visualizing the data in a chart.

4.6 Output

Here is the screenshot of the program's output:



```
04 3
06 1
07 1
09 2
10 3
11 6
14 1
15 2
16 4
17 2
18 1
19 1
```

Figure 4: Screenshot of the output

5 Problem 5: Point of Sale System

5.1 Problem Statement

The task is to create a point of sale system for a Frozen Yogurt Shop. The system should allow customers to order items (Frozen Yogurt, Ice Cream, Cookies, and Toppings), choose their sizes, containers, and toppings, and calculate the total order price. The system also keeps track of sales and generates a report for daily sales, including total revenue and the most popular items.

5.2 Approach

The approach for this point of sale system involves creating a menu of items, with prices depending on size and container type. The system asks the customer to make their selections and calculates the total price of the order based on their choices. The system allows multiple orders, and after each order, it updates the total sales, daily items sold, and customer-specific sales. At the end of the day, a report is generated that includes sales totals, customer order summaries, and item-wise sales analysis.

5.3 Code

```
def display_menu():
    print("\nWelcome to the Frozen Yogurt Shop!\n")
    print("Menu:")
    print("1. Frozen Yogurt")
    print("2. Ice Cream")
    print("3. Cookies")
    print("4. Toppings\n")

def get_price(item, size, container=None):
    prices = { "Frozen Yogurt": {"S": {"V": 3.09, "U": 3.19},"M": {"V": 3.89, "U": 3.99},
    "L": {"V": 4.69, "U": 4.79}, },"Ice Cream": {"S": {"V": 3.49, "U": 3.59},"M": {"V": 4.49,
    "U": 4.59},"L": {"V": 5.49, "U": 5.59},},"Toppings": {"G": 1.59, "N": 1.89},
    "Cookies": {"R": {"R": 1.49, "L": 2.49},"C": {"R": 1.59, "L": 2.59},},}

    if item in ["Frozen Yogurt", "Ice Cream"] and container:
        return prices[item][size.upper()][container.capitalize()]
    elif item == "Toppings":
        return prices[item][size.upper()]
    elif item == "Cookies" and container:
        return prices[item][container.upper()][size.upper()]
    return 0

def process_order(customers, total_sales, total_orders, daily_items, daily_totals):
    display_menu()
    name = input("Enter your name: ").strip().capitalize()
    if name in customers:
        print("Welcome back, {}!".format(name))
    else:
        print("Hello, {}! Nice to see you.".format(name))
        customers[name] = 0

    order_total = 0

    while True:
        item_choice = input("You want Frozen Yogurt or Ice Cream?
        (Y: Frozen Yogurt, I: Ice Cream): ").strip().upper()

        if item_choice == "Y":
            item = "Frozen Yogurt"
        elif item_choice == "I":
            item = "Ice Cream"
        else:
            print("Invalid choice. Try again.")
            continue

        size = input("What size? (S: Small, M: Medium, L: Large): ").strip().upper()
        container = input("What kind of container? (V: Cone or U: Cup): ").strip().capitalize
        order_total += get_price(item, size, container)

        topping_choice = input("Do you like to add a topping? (Y: Yes, N: No): ").strip().uppe
        if topping_choice == "Y":
            topping = input("What topping do you like? (G: Gummies, N: Nuts): ").strip().uppe
            order_total += get_price("Toppings", topping)

        cookie_choice = input("Do you like to have cookies? (Y: Yes, N: No): ").strip().upper
        if cookie_choice == "Y":
            cookie_type = input("What kind of cookies? (R: Raisin, C: Chocolate): ").strip().
            cookie_size = input("And the size? (R: Regular, L: Large): ").strip().upper()
            order_total += get_price("Cookies", cookie_size, cookie_type)

        more = input("Do you want to add another item? (Y: Yes, N: No): ").strip().upper()
        if more == "N":
            break

    if order_total > 0:
        print("Your order total is ${:.2f}".format(order_total))
        customers[name] += order_total
        total_sales += order_total
        total_orders += 1

        daily_items[item] += 1
        daily_totals[item] += order_total

    return customers, total_sales, total_orders, daily_items, daily_totals

def generate_report(customers, total_sales, total_orders, daily_items, daily_totals):
    print("\n*************** Sales Report ***********")
    print("Customer Name\tOrder Total")
    for customer, total in customers.items():
        print(f"{customer}\t${total:.2f}")

    print("\nItem\t\t#Orders\tOrder Total\tAverage per Order")
    for item, count in daily_items.items():
        if count > 0:
            avg = daily_totals[item] / count
            print(f"{item}\t{count}\t${daily_totals[item]:.2f}\t\t{avg:.2f}")

    print(f"\nGrand Total\t{total_orders}\t${total_sales:.2f}\t\t{total_sales / total_orders
    if total_orders > 0 else 0:.2f}")

customers = {}
total_sales = 0
total_orders = 0
daily_items = {"Frozen Yogurt": 0, "Ice Cream": 0}
daily_totals = {"Frozen Yogurt": 0, "Ice Cream": 0}

while True:
    customers, total_sales, total_orders, daily_items, daily_totals = process_order(
        customers, total_sales, total_orders, daily_items, daily_totals
    )
    next_action = input("Enter C to continue or Q to quit: ").strip().upper()
    if next_action == "Q":
        break
```

```
generate_report(customers, total_sales, total_orders, daily_items, daily_totals)
```

5.4 Explanation

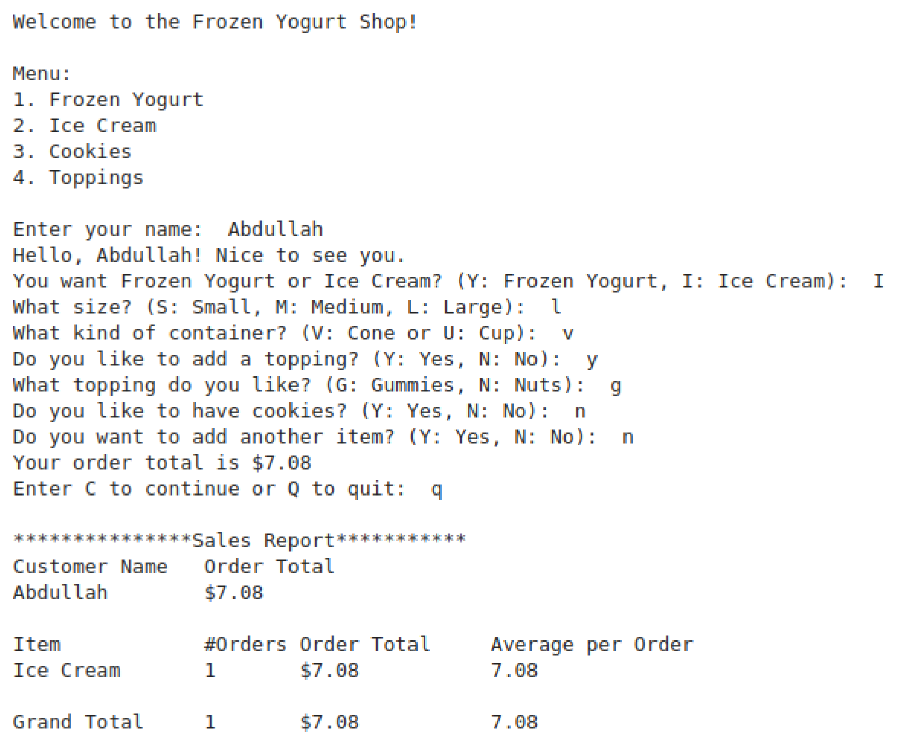
The point of sale system is designed to interact with the customer, allowing them to select items, choose their size, container type, toppings, and cookies. It calculates the total price of their order based on their choices. The program then records the sales, generates a daily report with item-wise sales and customer order totals, and displays the overall total revenue.
- **Menu**: The system displays a menu with options for Frozen Yogurt, Ice Cream, Cookies, and Toppings.
- **Order Input**: The customer selects an item and specifies size, container, and optional add-ons like toppings or cookies.
- **Pricing**: The price for each item is retrieved from a predefined pricing structure.
- **Sales Tracking**: After each order, the system updates the total sales, order count, and tracks individual customer purchases.

5.5 Conclusion

This point of sale system efficiently handles customer orders, calculates the total cost based on selections, and tracks sales data. It generates a detailed report for both individual customer orders and daily item sales, making it useful for shop owners to analyze performance and trends.

5.6 Output

Here is the screenshot of the program's output:



```
Welcome to the Frozen Yogurt Shop!

Menu:
1. Frozen Yogurt
2. Ice Cream
3. Cookies
4. Toppings

Enter your name:  Abdullah
Hello, Abdullah! Nice to see you.
You want Frozen Yogurt or Ice Cream? (Y: Frozen Yogurt, I: Ice Cream):  I
What size? (S: Small, M: Medium, L: Large):  l
What kind of container? (V: Cone or U: Cup):  v
Do you like to add a topping? (Y: Yes, N: No):  y
What topping do you like? (G: Gummies, N: Nuts):  g
Do you like to have cookies? (Y: Yes, N: No):  n
Do you want to add another item? (Y: Yes, N: No):  n
Your order total is $7.08
Enter C to continue or Q to quit:  q

***************Sales Report***********
Customer Name   Order Total
Abdullah        $7.08

Item            #Orders Order Total     Average per Order
Ice Cream       1       $7.08           7.08

Grand Total     1       $7.08           7.08
```

Figure 5: Screenshot of the output

6 Problem 6

6.1 Problem Statement

The objective of this project is to create a trivia game where users can add questions, view questions, play a game session by answering trivia questions, and keep track of scores in a leaderboard. The game should support multiple players and allow users to replay after a session.

6.2 Approach

To implement the trivia game, a structured approach was followed:

- **Data Storage:** Questions and their corresponding answers and point values are stored in a text file ("quest_load.txt"). The scores of players are maintained in another text file ("leaders.txt").
- **Loading Questions:** The function `load_questions()` reads from the question data file and stores them in a dictionary for easy access.
- **Adding New Questions:** Users can add new questions dynamically using the function `add_question()` which appends the new question to the file.
- **Displaying Questions and Scores:** Functions `show_questions()` and `show_scores()` allow users to view available questions and leaderboard rankings.
- **Game Session:** Players choose a number of questions to attempt, and responses are evaluated for correctness. Scores are updated accordingly in `play_game_session()`.
- **Replay Feature:** At the end of each session, players have the option to replay or exit.

6.3 Code

```
import random

QuestionDataFile= "quest_load.txt"
LeadersDataFile= "leaders.txt"
questions = {}
def load_questions():
    global questions
    try:
        with open(QuestionDataFile, "r") as file:
            questions = {}
            for line in file.readlines():
                question, answer, points = line.strip().split("|")
                questions[question] = (answer.lower(), int(points))
    except FileNotFoundError:
        questions = {}

def save_question(question, answer, points):
    with open(QuestionDataFile, "a") as file:
        file.write(f"{question}|{answer}|{points}\n")

def load_scores():
    try:
        with open(LeadersDataFile, "r") as file:
            return {line.split("|")[0]: int(line.split("|")[1]) for line in file.readlines()}
    except FileNotFoundError:
        return {}

def save_scores(scores):
    with open(LeadersDataFile, "w") as file:
        for player, score in scores.items():
            file.write(f"{player}|{score}\n")

def add_question():
    question = input("Enter␣the␣question:␣").strip()
    answer = input("Enter␣the␣answer:␣").strip().lower()
    points = input("Enter␣the␣point␣value:␣").strip()
    save_question(question, answer, points)
    print("Question␣added␣successfully!")

def show_questions():
    if not questions:
        print("No␣questions␣available.")
    else:
        index = 1
        for q, (a, p) in questions.items():
            print(f"{index}.␣{q}␣(Points:␣{p})")
            index += 1

def show_scores():
    scores = load_scores()
    if not scores:
        print("No␣scores␣available.")
    else:
        sorted_scores = sorted(scores.keys(), key=scores.get, reverse=True)
        print("<<<␣LEADER␣BOARD␣>>>")
        for player in sorted_scores:
            print(f"{player.capitalize():<15}␣{scores[player]}")
def play_game_session(name, session_questions):

    scores = load_scores()

    if name in scores:
        print(f"Welcome␣back␣{name.capitalize()}!␣Your␣previous␣score␣was␣{scores[name]}")
    else:
        print("Welcome␣to␣Trivia␣Madness!")

    while True:
        if not session_questions:
            print("No␣more␣questions␣left␣in␣this␣session.")
            return
        try:
            num_questions = int(input(f"How␣many␣questions␣would␣you␣like␣to␣attempt?
␣␣␣␣␣␣␣␣␣␣␣␣(Max␣{len(session_questions)}):␣"))
            if 1 <= num_questions <= len(session_questions):
                break
            else:
                print(f"Please␣enter␣a␣number␣between␣1␣and␣{len(session_questions)}.")
        except ValueError:
            print("Invalid␣input.␣Please␣enter␣a␣valid␣number.")
    selected_questions = random.sample(list(session_questions.items()), num_questions)
    score = scores.get(name, 0)
    for q, (a, p) in selected_questions:
        answer = input(f"{q}␣").strip().lower()
        if answer == a:
            print("Correct!")
            score += p
        else:
            print(f"Incorrect Response! The correct answer was: {a}")
        del session_questions[q]
    scores[name] = score
    save_scores(scores)
    print(f"Game over! {name.capitalize()} scored {score} points.")
    while True:
        replay = input("Do you want to play again? Y to play, Q to quit: ").strip().lower()
        if replay == "y":
            play_game_session(name, session_questions)  # Continue with remaining questions
            break
        elif replay == "q":
            break
        else:
            print("Invalid choice. Please enter Y or Q.")

def play_game(name):
    if not questions:
        print("No questions have been loaded. Please load questions first.")
        return
    session_questions = questions.copy()

    play_game_session(name, session_questions)

load_questions()
while True:
    print("\nPlease select an item from the menu below:")
    print("1– Load Questions")
    print("2– Add Question")
    print("3– Show Questions")
    print("4– Play Game")
    print("5– Show Scores")
    print("6– Quit")

    choice = input("Enter your choice: ")

    if choice == "1":
        load_questions()  # Reload questions from file
        print("Questions loaded successfully!")
    elif choice == "2":
        add_question()
    elif choice == "3":
        show_questions()
    elif choice == "4":
        name = input("Enter your first name: ").strip().lower()
        play_game(name)
    elif choice == "5":
        show_scores()
    elif choice == "6":
        print("Exiting game. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

6.4 Explanation

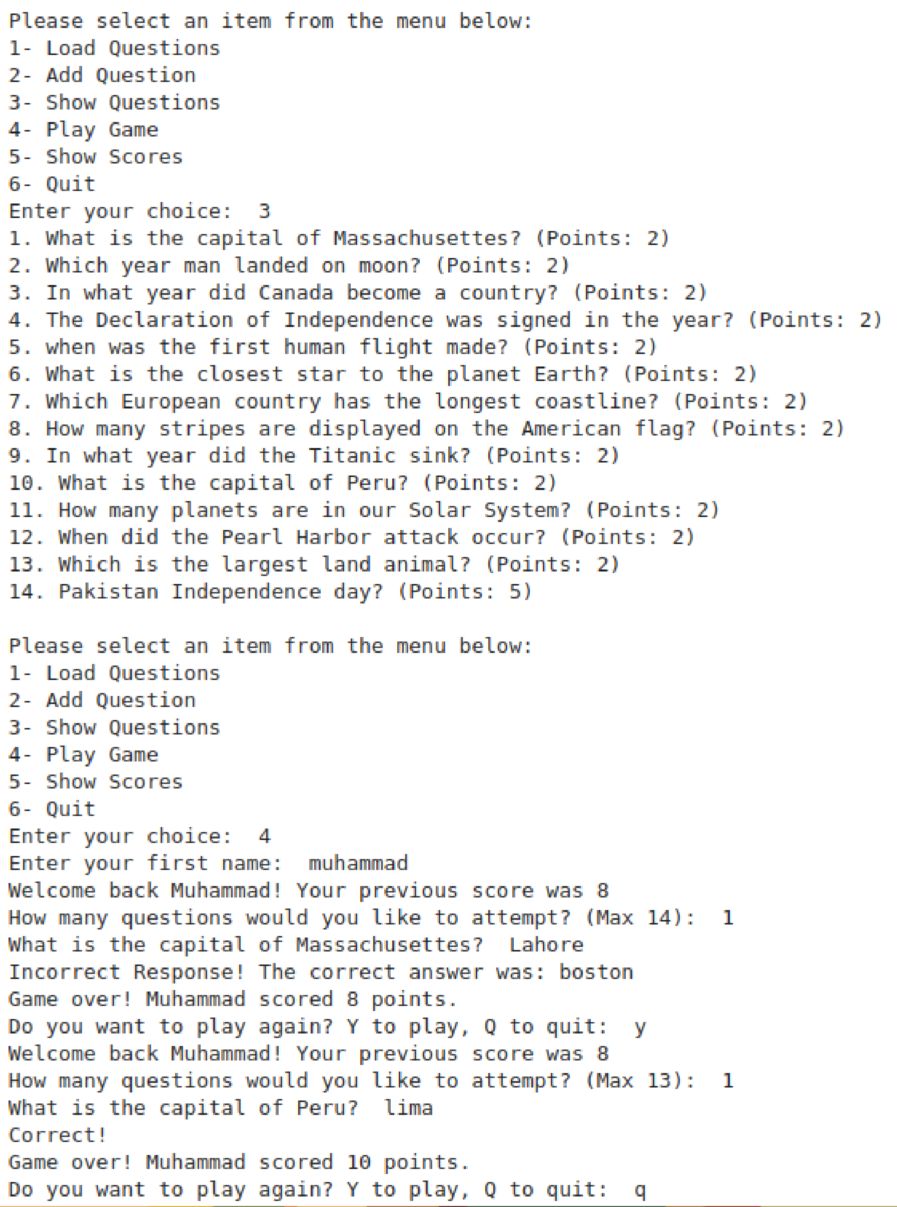
The program begins by loading questions and scores from their respective files. Players can add questions, view available questions, and start a game session. During a session, a specified number of questions are randomly selected. The player inputs their answers, and the system checks for correctness. Points are awarded accordingly, and the updated score is saved. A replay feature allows users to restart the session with remaining questions. The game continues until the player chooses to quit.

6.5 Conclusion

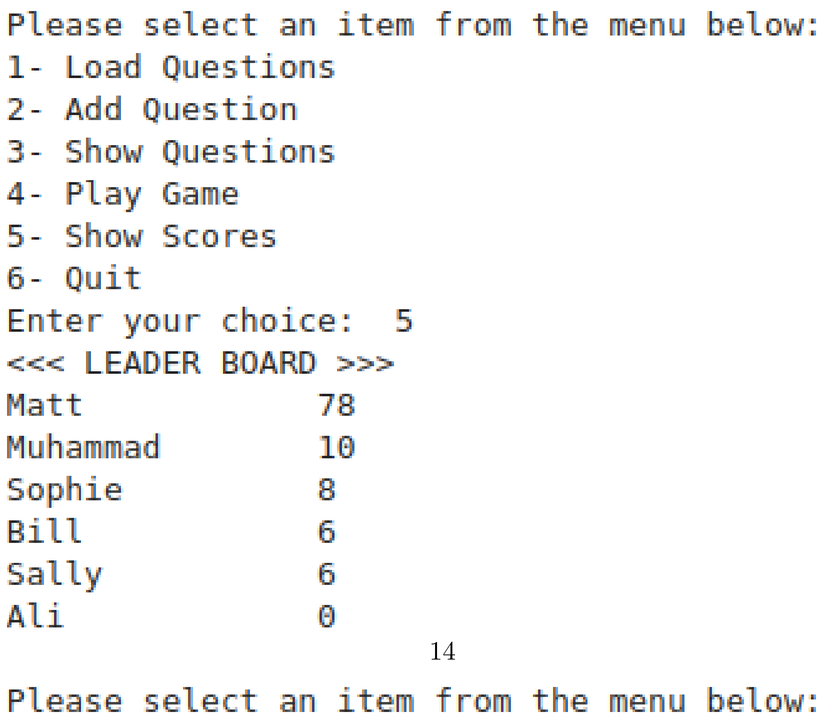
This trivia game successfully provides an interactive way for users to test their knowledge. The implementation includes features for question management, score tracking, and game replay.

6.6 Output

Here is the screenshot of the program's output:



```
Please select an item from the menu below:
1- Load Questions
2- Add Question
3- Show Questions
4- Play Game
5- Show Scores
6- Quit
Enter your choice:  3
1. What is the capital of Massachusettes? (Points: 2)
2. Which year man landed on moon? (Points: 2)
3. In what year did Canada become a country? (Points: 2)
4. The Declaration of Independence was signed in the year? (Points: 2)
5. when was the first human flight made? (Points: 2)
6. What is the closest star to the planet Earth? (Points: 2)
7. Which European country has the longest coastline? (Points: 2)
8. How many stripes are displayed on the American flag? (Points: 2)
9. In what year did the Titanic sink? (Points: 2)
10. What is the capital of Peru? (Points: 2)
11. How many planets are in our Solar System? (Points: 2)
12. When did the Pearl Harbor attack occur? (Points: 2)
13. Which is the largest land animal? (Points: 2)
14. Pakistan Independence day? (Points: 5)

Please select an item from the menu below:
1- Load Questions
2- Add Question
3- Show Questions
4- Play Game
5- Show Scores
6- Quit
Enter your choice:  4
Enter your first name:  muhammad
Welcome back Muhammad! Your previous score was 8
How many questions would you like to attempt? (Max 14):  1
What is the capital of Massachusettes?  Lahore
Incorrect Response! The correct answer was: boston
Game over! Muhammad scored 8 points.
Do you want to play again? Y to play, Q to quit:  y
Welcome back Muhammad! Your previous score was 8
How many questions would you like to attempt? (Max 13):  1
What is the capital of Peru?  lima
Correct!
Game over! Muhammad scored 10 points.
Do you want to play again? Y to play, Q to quit:  q
```



```
Please select an item from the menu below:
1- Load Questions
2- Add Question
3- Show Questions
4- Play Game
5- Show Scores
6- Quit
Enter your choice:  5
<<< LEADER BOARD >>>
Matt            78
Muhammad        10
Sophie          8
Bill            6
Sally           6
Ali             0
```

```
Please select an item from the menu below:
```