#Name:-ABDUL RAYYAAN.A.R

#Reg.no:192210532

#Question 1:

# Function to read questions from Questions.txt

def read\_questions(file\_name):

questions = []

with open(file\_name, 'r') as file:

lines = file.readlines()

current\_question = None

for line in lines:

line = line.strip()

if line:

if line.startswith('Q:'):

if current\_question:

questions.append(current\_question)

current\_question = {"question": line[2:], "options": []}

elif line.startswith('A:'):

current\_question["options"].append(line[2:])

if current\_question:

questions.append(current\_question)

return questions

# Function to read correct answers from Answers.txt

def read\_answers(file\_name):

answers = []

with open(file\_name, 'r') as file:

answers = [line.strip() for line in file]

return answers

# Function to display quiz questions and options

def display\_quiz(questions):

for i, question in enumerate(questions):

print(f"{i + 1}. {question['question']}")

for j, option in enumerate(question['options']):

print(f" {chr(ord('A') + j)}. {option}")

print()

# Function to display correct answers

def display\_answers(questions, answers):

print("Correct Answers:")

for i, (question, correct\_answer) in enumerate(zip(questions, answers)):

print(f"{i + 1}. {question['question']}")

print(f" Correct Answer: {correct\_answer}")

print()

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

questions\_file = "Questions.txt"

answers\_file = "Answers.txt"

quiz\_questions = read\_questions(questions\_file)

correct\_answers = read\_answers(answers\_file)

if len(quiz\_questions) != len(correct\_answers):

print("Error: Number of questions and answers do not match.")

else:

print("Welcome to the Quiz!")

display\_quiz(quiz\_questions)

input("Press Enter to see the correct answers...")

display\_answers(quiz\_questions, correct\_answers)

#Question 2:

# Define custom exceptions

class ValueTooLarge(Exception):

pass

class ValueTooSmall(Exception):

pass

# Function to play the number guessing game

def play\_game(target\_number):

while True:

try:

user\_guess = int(input("Guess the number: "))

if user\_guess > target\_number:

raise ValueTooLarge

elif user\_guess < target\_number:

raise ValueTooSmall

else:

print("Congratulations! You guessed the correct number.")

break

except ValueError:

print("Invalid input. Please enter a valid number.")

except ValueTooLarge:

print("Too large! Try again.")

except ValueTooSmall:

print("Too small! Try again.")

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

try:

target\_number = int(input("Enter the number to be guessed: "))

play\_game(target\_number)

except ValueError:

print("Invalid input. Please enter a valid number as the target.")

#Sample output:

Enter the number to be guessed: 3

Guess the number: 3

Congratulations! You guessed the correct number.

#Question 3:

d = {"R": 0, "G": 255, "B": 0, "other": {"opacity": 0.6}}

# (a) d["R"]

# Return value: 0

# Final value of d: {"R": 0, "G": 255, "B": 0, "other": {"opacity": 0.6}}

result\_a = d["R"]

# (b) d.pop("R")

# Return value: 0

# Final value of d: {"G": 255, "B": 0, "other": {"opacity": 0.6}}

result\_b = d.pop("R")

# (c) d["R"] = 255

# Return value: None (assignment)

# Final value of d: {"G": 255, "B": 0, "R": 255, "other": {"opacity": 0.6}}

result\_c = d["R"] = 255

# (d) d["H"]

# This raises a KeyError because "H" is not in the dictionary, and d remains unchanged.

try:

result\_d = d["H"]

except KeyError as e:

result\_d = str(e)

# (e) d.keys()

# Return value: dict\_keys(['R', 'G', 'B', 'other'])

# Final value of d: {"G": 255, "B": 0, "R": 255, "other": {"opacity": 0.6}}

result\_e = d.keys()

# (f) d["other"]["blur"] = 0.1

# Return value: None (assignment)

# Final value of d: {"R": 0, "G": 255, "B": 0, "other": {"opacity": 0.6, "blur": 0.1}}

result\_f = d["other"]["blur"] = 0.1

# (g) d[["H","S","L"]] = [120,98,5]

# This raises a TypeError because a list cannot be used as a dictionary key, and d remains unchanged.

try:

result\_g = d[["H", "S", "L"]] = [120, 98, 5]

except TypeError as e:

result\_g = str(e)

# (h) d["R","B","G"]

# This raises a TypeError because a tuple cannot be used as a dictionary key directly, and d remains unchanged.

try:

result\_h = d["R", "B", "G"]

except TypeError as e:

result\_h = str(e)

# Printing the results

print("(a) d[\"R\"]: ", result\_a)

print("(b) d.pop(\"R\"): ", result\_b)

print("(c) d[\"R\"] = 255: ", result\_c)

print("(d) d[\"H\"]: ", result\_d)

print("(e) d.keys(): ", list(result\_e))

print("(f) d[\"other\"][\"blur\"] = 0.1: ", result\_f)

print("(g) d[\"H\",\"S\",\"L\"] = [120,98,5]: ", result\_g)

print("(h) d[\"R\",\"B\",\"G\"]: ", result\_h)

#Question 4:

# Define the questions and answers as separate lists

questions = ["What is the capital of France?", "Who wrote the play 'Romeo and Juliet'?", "What is the largest planet in our solar system?"]

answers = ["Paris", "William Shakespeare", "Jupiter"]

# Create a quiz function using zip

def create\_quiz(questions, answers):

score = 0

# Combine questions and answers using zip

quiz = zip(questions, answers)

for question, correct\_answer in quiz:

user\_answer = input(f"{question}\nYour answer: ").strip()

if user\_answer.lower() == correct\_answer.lower():

print("Correct!\n")

score += 1

else:

print(f"Wrong! The correct answer is {correct\_answer}.\n")

print(f"Quiz completed! Your score: {score}/{len(questions)}")

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

print("Welcome to the Quiz!")

create\_quiz(questions, answers)

#Sample output:

Welcome to the Quiz!

What is the capital of France?

Your answer: paris

Correct!

Who wrote the play 'Romeo and Juliet'?

Your answer: William Shakespeare

Correct!

What is the largest planet in our solar system?

Your answer: Jupiter

Correct!

Quiz completed! Your score: 3/3

#Question 5:

# Create a tuple

my\_tuple = (1, 2, 3, 4, 5, 6, 7, 8)

# Get the 4th element (index 3)

fourth\_element = my\_tuple[3]

# Get the 4th element from the end (index -4)

fourth\_from\_end = my\_tuple[-4]

# Print the results

print("4th element:", fourth\_element)

print("4th element from the end:", fourth\_from\_end)

#Sample output:

4th element: 4

4th element from the end: 5