INFS1201-Computer Program

Assignment

**Read the instructions carefully and follow them exactly**

|  |  |
| --- | --- |
| Assignment Weight | 5% of your course grade |
| Due Date and Time | 11:59 PM on Saturday, March 2, 2024 |
| Important | As outlined in the syllabus, late submissions will not be accepted. |
| Any files with syntax errors will automatically be excluded from  grading. Be sure to test your code before you submit it. |
| For all functions, **make sure you have written good docstrings** **that include parameters and return types, function description,** and preconditions if any. |

# **Assignment Guidelines**

**Academic Integrity and Individual Work:**

* This assignment is an individual task, and academic integrity is of utmost importance. Please review the Plagiarism and Academic Integrity policy presented in the class.

**Assignment Objectives:**

* The primary goal of this assignment is to reinforce and practice the concepts covered in the course, including strings (including indexing, slicing, and string methods), lists, tuples, control structures (if statements and loops), the use of the range function, function design, and function calls.

**Allowed Data Structures:**

* You may use only strings, tuples, and lists as data structures in your solution. **Usage of any other collections like sets or dictionaries is not permitted**. Implementing solutions involving unauthorized collections will result in those questions not being graded.

**Comprehensions and Inline Statements:**

* Comprehensions (list comprehensions, dictionary comprehensions, etc.) are not allowed. You should use conventional loops and constructs to solve the problems.
* The use of inline for and inline if statements in list, tuple, or dictionary creation is also not allowed. Solutions should use conventional loop structures for iteration and conditional statements.
* Failure to adhere to the restrictions on comprehension and inline statements will result in a grade of zero for the respective questions.

**Multiple Submissions:**

* You have the option to make multiple submissions; however, only the last submission before the deadline will be graded.

**Submission Format:**

* You should submit only one Python file containing your solution.
* Name your files as 'q1.py,' 'q2.py,' and so forth.
* Archive the files in a zip. Name the file as "xxxxxxxx," replacing "xxxxxxxx" with your student number.
* Upload .zip file to Dropbox.
* Each file should exclusively contain the necessary functions and imports.
* While testing, you may include a main program within the same file. However, please remove any excess code before submitting.
* Your submission will be automatically evaluated using a test script.

**Syntax Errors:**

* Ensure that your programs run without syntax errors. The instructor will run your code during grading, and any syntax errors will lead to a zero grade for the question.

**Testing Your Code:**

* Each part of your solution should be thoroughly tested. While your program may not always produce the correct answer for the provided test cases, it should not produce any Python runtime errors, otherwise, that question will be marked with zero points. Ensure that your program is well-tested with additional examples beyond the provided ones.

**Citing Sources:**

* If you seek help or inspiration from the Internet or another person, it is essential to properly cite the code sections you borrow and the source. Failure to do so will result in a grade of zero for the entire assignment. Keep in mind that borrowing code should not exceed 15% of your solution.

**Global Variables:**

* Global variables are not allowed.  Failure to comply with this rule will result in a 10% reduction in your overall grade.

**Test Cases:**

* Before submitting the file, ensure that all sample run code, including test cases and example method calls, is removed from the Python file. Submitted files should contain only the solution code.

****

**Assignment: Qatar Asia Cup Match System 2023**

Objective: The objective of this assignment is to design and implement a system for managing and analyzing match data for the Qatar Asia Cups. Participants will engage in developing functions to calculate match-up distances between ticket numbers, predict match winners based on historical performance data, input match data accurately, identify top goal scorers, and determine the highest-scoring team. This assignment aims to assess students’ proficiency in working with various data structures, including lists and tuples, and implementing logical algorithms to derive meaningful insights from the provided match data. Students will demonstrate their coding skills in Python to enhance the overall experience and excitement of the Qatar Asia Cups.

**Problem 1: Qatar Asia Cups Match-Up**

Get ready for the Qatar Asia Cup, where the excitement is building, and football fans eagerly anticipate the thrill of the game and the chance to win big with their ticket numbers! In this challenge, football enthusiasts have a shot at fabulous prizes not only with a perfect match but also for being close to the winning ticket.

Your mission is to calculate the "match-up distance" between two ticket numbers, a metric that determines the prize a fan will receive. The match-up distance is the number of positions where the two ticket numbers differ.

Consider the following scenario:

**Example:**

* Winning Ticket Number: "A1B2"
* Fan's Ticket Number: "AAB2"

Here, the match-up distance is 1, as the strings differ at exactly one position.

Your task is to create a function that computes the match-up distance between the winning ticket number and the fan's ticket number. Afterward, the function should return a tuple representing the match-up distance, both ticket numbers with matched parts replaced by '\*', and the announcement for the prize earned by the fan.

* A match-up distance of 0 signifies the fan has won the grand prize!
* A match-up distance of 1 or 2 indicates the fan has won a smaller prize.
* A match-up distance exceeding 2 means the fan hasn't won any prizes. :(

**Prize Scheme:**

* Grand Prize (Match-up distance of 0): A VIP pass to enjoy the Qatar Asia Cups from the best seats in the stadium!
* Small Prize (Match-up distance of 1 or 2): An exclusive Qatar Asia Cups-themed football jersey.
* No Prize (Match-up distance exceeding 2): Better luck next time!

If the two ticket numbers are not of the same length, return 'None'.

|  |
| --- |
| **Test Cases:**  >>> checkMatchUpDistance("Q123456", "Q123456")  (0, "Q123456", "\*\*\*\*\*\*\*", "Your ticket number has a match-up distance of 0! Congratulations, you are the winner of the Grand Prize: A VIP pass to enjoy the Qatar Asia Cups from the best seats in the stadium!")  >>> checkMatchUpDistance("Q173456", "Q123456")  (1, "Q173456", "\*\*2\*\*\*\*", "Your ticket number has a match-up distance of 1! Congratulations, you win a small prize: An exclusive Qatar Asia Cups-themed football jersey.")  >>> checkMatchUpDistance("Q123456", "Q127890")  (4, "Q123456", "\*\*\*7890", "Your ticket number has a match-up distance of 4! Better luck next time!")  >>> checkMatchUpDistance("Q023456", "Q12345678")  None |

**Problem 2: Qatar Asia Cup Match Predictor**



You are developing a match result prediction system for the Qatar Asia Cup. The system aims to predict the winner of a match based on the historical performance data of two teams. The historical performance data is now represented by a list for each match, containing the names of the two teams and a list with the number of wins, losses, and draws.

team\_data = [

['Qatar', 'India', ['wins', 10, 'losses', 5, 'draws',3]],

['Jordan', 'Iraq', ['wins',4, 'losses', 3, 'draws', 2]],

# ... additional match data

]

The match data for a single entry in the **team\_data** list looks like this:

['Qatar', 'India', ['wins', 10, 'losses', 5, 'draws', 3]]

In this example:

* The first element is the name of the first team, which is 'Qatar'.
* The second element is the name of the second team, which is 'India'.
* The third element is a list containing performance statistics for the match:
  + 'wins', 10 (indicating that the first team, 'Qatar', has won 10 matches)
  + 'losses', 5 (indicating that the first team, 'Qatar', has lost 5 matches)
  + 'draws', 3 (indicating that the first team, 'Qatar', has drawn 3 matches)

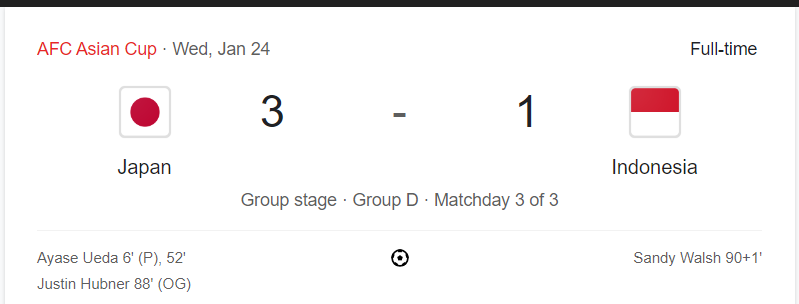
Implement a function **predictMatchWinner** that takes the historical performance data of a match and predicts the winner based on the following rules:

* If one team has more wins than the other, that team is predicted to be the winner.
* If both teams have equal wins, the prediction is inconclusive (return **None**).

The function should search for the statistics in the list and return the name of the predicted winning team.

|  |
| --- |
| **Test Cases:**  team\_data = [  ['Qatar', 'India', ['wins', 10, 'losses', 5, 'draws', 3]],  ['Jordan', 'Iraq', ['wins', 3, 'losses', 3, 'draws', 2]],  ['Pakistan', 'China', ['wins', 6, 'losses', 8, 'draws', 4]]  ]  predicted\_winner = predictMatchWinner('Qatar', 'India', team\_data)  print(predicted\_winner)  # Expected output: 'Qatar' (Qatar has more wins than India)  predicted\_winner = predictMatchWinner('india', 'qatar', team\_data)  print(predicted\_winner)  # Expected output: 'Qatar' (Qatar has more wins than India)  predicted\_winner = predictMatchWinner('Jordan', 'Iraq', team\_data)  print(predicted\_winner)  # Expected output: None (Jordan and Iraq have the same number of wins)  predicted\_winner = predictMatchWinner('Pakistan', 'China', team\_data)  print(predicted\_winner)  # Expected output: 'China' (China has more wins than Pakistan) |

**Problem 3: Match Data Input**



Create a function named **inputMatchData** that printsa 2D list **matchData**. The function should prompt the user to input the number of matches **numMatches** and information for each match, including:

* Teams participating in each match.
* Final scores of each match.
* Goals for each match.
* Names of players who scored goals for each team.

Ensure the data is accurately formatted and add it to the **matchData** list.

Here's an illustrative sample structure for the **matchData** list:

[

['Qatar', 'Iraq', 3, 2, ['Hassan Al-Haydos', 'Almoez Ali', 'Akram Afif'],['Amir Al-Ammari', 'Hussein Ali']],

['Jordan', 'Qatar', 1, 1, ['Mahmoud Al-Mardi'], ['Akram Afif']]

# ... additional match data

]

This structure captures teams, scores, and a list detailing goals for each team involved in the match.

For example, the input **[['Qatar', 'Iraq', 3, 2, ['Hassan Al-Haydos', 'Almoez Ali', 'Akram Afif'], ['Amir Al-Ammari', 'Hussein Ali']], ['Jordan', 'Qatar', 1, 1, ['Mahmoud Al-Mardi'], ['Akram Afif']]]** represents two matches:

1. The first match is between 'Qatar' and 'Iraq' with the final scores being 'Qatar' 3 - 2 'Iraq'. Goals scored by 'Hassan Al-Haydos', 'Almoez Ali', 'Akram Afif' for Qatar, and goals scored by 'Amir Al-Ammari', 'Hussein Ali' for Iraq.
2. The second match is between 'Jordan' and 'Qatar' with the final scores being 'Jordan' 1 - 1 'Qatar'. Goals scored by 'Mahmoud Al-Mardi' for Jordan and a goal scored by 'Akram Afif' for Qatar.

**Use the following input for testing the function:**

Enter the number of matches: 2

Enter the teams playing in the match (separated by comma): Qatar,Iraq

Enter the final scores of the match (separated by space): 3 2

Enter goal #1 for Qatar: hassan al-haydos

Enter goal #2 for Qatar: Almoez Ali

Enter goal #3 for Qatar: Akram Afif

Enter goal #1 for Iraq: Amir Al-Ammari

Enter goal #2 for Iraq: Hussein Ali

Enter the teams playing in the match (separated by comma): Jordan, qatar

Enter the final scores of the match (separated by space): 1 1

Enter goal #1 for Jordan: mahmoud Al-Mardi

Enter goal #2 for Qatar: Akram Afif

[['Qatar', 'Iraq', 3, 2, ['Hassan Al-Haydos', 'Almoez Ali', 'Akram Afif'], ['Amir Al-Ammari', 'Hussein Ali']], ['Jordan', 'Qatar', 1, 1, ['Mahmoud Al-Mardi'], ['Akram Afif']]]

**Problem 4: Qatar Asia Cup Team Points Calculator**



You are developing a program to calculate the points for each football team participating in the Qatar Asia Cup. The points are awarded based on match results, considering wins, draws, and losses.

Here's an illustrative sample structure for the **matchData** list:  
 [

['Qatar', 'Iraq', 3, 2],

['Jordan', 'Qatar', 1, 1]

# ... additional match data

]

For example, the input **[['Qatar', 'Iraq', 3, 2], ['Jordan', 'Qatar', 1, 1]]** represents two matches:

1. The first match is between 'Qatar' and 'Iraq' with the final scores being 'Qatar' 3 - 2 'Iraq'.
2. The second match is between 'Jordan' and 'Qatar' with the draw final scores being 'Jordan' 1 - 1 'Qatar'.

The points are awarded as follows:

* A team receives 3 points for a win.
* A team receives 1 point for a draw.
* A team receives 0 points for a loss.

Write a function **calculateTeamPoints** that takes a list of match results (**matchData**) as input and returns a 2D list containing team names and their total points.

|  |
| --- |
| **#Test cases**  # Sample Run  matchData = [  ['Qatar', 'Iraq', 3, 2],  ['Jordan', 'Qatar', 1, 1],  ['Jordan', 'Palestine', 1, 2]  ]  team\_points = calculateTeamPoints(matchData)  print(team\_points)  # Expected output: [['Qatar', 4], ['Iraq', 0], ['Jordan', 1], ['Palestine', 3]] |

Ensure your function correctly calculates the total points for each team based on the provided match data and returns a 2D list with team names and their corresponding total points.

**Problem 5: Top Goal Scorers**

Write a function named **topGoalScorers** that takes a 1D list **goalData** as input and returns a list of tuples containing only the names of the top goal scorers, their respective goal counts, and the associated country name.

The **goalData** is a 1D list where each group of three consecutive elements represents:

* Team Name
* Player Name
* Number of Goals Scored

The **goalData** follows this structure:

['Qatar', 'Hassan Al-Haydos', 3, 'Qatar', 'Akram Afif', 1, 'Iraq', 'Amir Al-Ammari', 2, 'Iraq', 'Hussein Ali', 1, 'Jordan', 'Mahmoud Al-Mardi', 2, 'Jordan', 'Musa Al-Taamari', 1, 'Palestine', 'Oday Dabbagh', 3, 'Palestine', 'Bader Nasser', 1]

Write the **topGoalScorers** function to calculate and return a list of tuples, each containing only the name of a top goal scorer, their respective goal count, and the associated country name. The list should only include entries for the top goal scorers and be sorted in descending order based on the goal counts.

Here are test cases to verify your function:

|  |
| --- |
| **#Test cases**  # Sample Run  goalData = ['Qatar', 'Hassan Al-Haydos', 3, 'Qatar', 'Akram Afif', 1, 'Iraq', 'Amir Al-Ammari', 2, 'Iraq', 'Hussein Ali', 1, 'Jordan', 'Mahmoud Al-Mardi', 2, 'Jordan', 'Musa Al-Taamari', 1, 'Palestine', 'Oday Dabbagh', 3, 'Palestine', 'Bader Nasser', 1]  print(topGoalScorers(goalData))  # Expected output: [('Qatar', 'Hassan Al-Haydos', 3), ('Palestine', 'Oday Dabbagh', 3)] |

Ensure the function correctly identifies and includes only the top goal scorers with their country names based on their goal counts.

**Problem 6: Highest-Scoring Team**

Write a function named **highestScoringTeam** that takes **matchData** as input and returns a tuple containing the highest total goals scored and the name of the team achieving that score.

The **matchData** is a list of tuples that follows the structure:

[

('Qatar', 'Iraq', 3, 2),

('Jordan', 'Qatar', 1, 1)

# ... additional match data

]

Each inner tuple represents team a match.

Write the highestScoringTeam function to determine and return a tuple with the highest total goals scored and the name of the team achieving that score.

Here are test cases to verify your function:

|  |
| --- |
| **#Test cases 1**  matchData = [  ('Qatar', 'Iraq', 3, 2),  ('Jordan', 'Qatar', 1, 1)  ]  print(highestScoringTeam(matchData))  # Expected output: (4, ['Qatar']) |

Ensure the function handles cases where multiple teams have the same highest total goals scored, returning all of them.

|  |
| --- |
| **#Test cases 2**  # Sample Run  matchData = [  ('Qatar', 'Iraq', 3, 2),  ('Jordan', 'Qatar', 1, 1),  ('Jordan', 'Palestine', 3, 1)  ]  result = highestScoringTeam(matchData)  print(result)  # Expected output: (4, ['Qatar', 'Jordan']) |