CSCI 3901 Lab 3: Testing

Team Members:

Name	Banner ID	Email ID
Rushil Borad	B00977837	rs519505@dal.ca
Jay Sanjaybhai Patel	B00982253	jy451478@dal.ca

Part 1

Ambiguities and Clarifications:

- The problem statement does not specify how teams with no recorded games should be considered in the leaderboard creation process.
    - Clarification: The Teams should be mentioned in the leaderboard with their minimal matches and scores i.e. 0.
- What will the leaderboard display when there is no game outcome recorded?
    - Clarification: In this case only the header row of the leaderboard will be printed.
- There's no information on how frequently the leaderboard will be updated, leaving uncertainty about timing and frequency requirements.
    - Clarification: The leaderboard will be updated after recording every match or whenever the function is called.
- What is the maximum number of scores that can be recorded in a match for each team.
    - Clarification: As per the problem there are 4 columns allotted in the leaderboard for points scored by the team. Thus, we are assuming that the maximum score for a team can be 9999.
- What is the maximum number of games won or lost.
    - Clarification: As mentioned in the problem there are 2 columns allotted to showcase the number of games lost or won by any team on the leaderboard. Thus, we are assuming the maximum games won or lost can be 99.
- When comparing the team’s name is already present should their case be considered or not.
    - Clarification: When comparing the team's name, it should be considered as case insensitive.

Boundary conditions:

Maximum Capacity of Teams:

- The league has space for a maximum of 24 teams.
- This sets the upper limit on the number of teams that can participate in the league.
- Any attempt to add more than 24 teams should be handled appropriately.

Column Width in Leaderboard:
The description specifies the width of each column in the leaderboard output:

- Team name: 15 columns
- Number of games won: 2 columns.
- Number of games lost: 2 columns.
- Number of games tied: 2 columns.
- Points scored by the team: 4 columns.
- Points scored against the team: 4 columns.
- Each column is separated by a space.

Ensuring that the output matches to specified column widths is critical for readability and alignment.

Maximum Scores and Outcomes:

- There are boundary conditions related to the maximum scores and outcomes achieved by teams, including:
    - Maximum number of games won by a team.
    - Maximum number of games lost by a team.
    - Maximum number of games tied by a team.
    - Maximum points scored by a team in total.
    - Maximum points scored against a team in total.

Outline of the control flow

Adding Teams to the League:

- Control flow:
    - Check if the team is already in the league.
    - If not, add the team to the league.
    - Return true if the team was successfully added, false otherwise.

Recording Game Outcomes:

- Control flow:
    - Record the outcome of the game between team1 and team2 with scores scoreTeam1 and scoreTeam2.
    - Return true if the game outcome was successfully recorded, false otherwise.

Generating Leaderboard:

- Control flow:
    - Calculate the standings of all teams based on game outcomes, following the specified tie-breaking rules.
    - Sort the teams in descending order of success based on the tie-breaking rules.
    - Format the leaderboard output according to the specified column widths and header row.
    - Return the string representation of the leaderboard.

Normal Order of Method Invocation:

- Add Team
    - Teams are added to the league.
- Record Game Outcome:
    - Games are played between teams, and outcomes are recorded.
- Create Leaderboard
    - The leaderboard is generated based on the recorded game outcomes, following the specified tie-breaking rules.

Part 2 – Test Cases

Method: public boolean addTeam(String teamName) :

Input Validation:
- Provide null for the team's name.
- Provide an empty string as a team name.

Boundary Cases
- Adding the first team to the league
- Adding the last team to fill the league.
- Adding 25th team to the league

Control Flow
- Add a team name with string length of more than 15.
- Add a new team successfully.
- Add a team and verify that it exists in the league.
- Add a team name with leading/trailing whitespaces.
- Add a team name containing special characters.

Data Flow
- Call before recordGameOutcome()
- Call before createLeaderBoard()

Test	**Outcome**	**Note**
Null Input	false	Expecting a team name but nothing there
Empty Input	false	Team name length must be greater than 0
Ex - "Seraphina Montgomery"	false	Team name length must be less than 15
Adding two teams "Team C" and "Team C"	false	Team name must be unique
Adding two teams "Team A" and "Team a"	false	Consider the team's name as case insensitive.
Adding two teams "Team A" and "TeamA"	false	Compare the team after removing all the in between spaces if it had.
Adding two teams "Team A" and "Team A"	false	Compare team after removing trailing spaces
"Team@!$#"	true	Special Characters and integers are accepted as a string
Add teams until reaching the maximum team capacity (24 Teams)	true	25th team is not accepted
"Team α"	true	Unicode

Method: public boolean recordGameOutcome(String team1, String team2, int scoreTeam1, int scoreTeam2)

Input Validation Tests:

- Null values for team1 and team2
- Empty strings for team1 and team2

Boundary Cases

- Recording the first game
- Recording the last game before reaching any limit
- Negative values for scoreTeam1 and/or scoreTeam2

Control Flow

- Same team name for both team1 and team2
- Non-existent team names for team1 and/or team2
- Recording a Game with Team1 winning
- Recording a Game with Team2 winning
- Recording a Tied Game
- Game won or lost exceeding the maximum allowed integer value i.e. 99. (two columns)
- Scores of the team exceeding the maximum allowed integer value i.e. 9999. (four columns)

Data Flow

- Call after addTeam()
- Call before createLeaderBoard( )

Test	**Outcome**	**Note**
null, “Salty“, 15, 12	false	Neither team1 nor team2 has its name as null
“Anchor“, ““, 15, 12	false	Neither team1 nor team2 has its name as empty string
“Anchor“, “Salty“, 15, -1	false	Game score must be greater than zero
“Anchor“, “Salty“, 99999, 12	false	Game score must be less than 10000
“Anchor“, “Anchor“, 15, 12	false	Two team name must not be same
“Anchor“, “Salty“, 0, 0	true	Both teams may have the same score and it may be zero and greater than that 0.

Method: public String createLeaderBoard()

Input Validation Tests:
No Input
Boundary Cases

- Print leaderboard with only one game recorded.
- Print Maximum number of teams, games and scores recorded.

Control Flow

- If no games recorded, then print header row only.
- Check if multiple games recorded with different outcomes (like tied, won or loss)
- For Tie breaker check the following conditions:
    - Teams with varying numbers of games won.
        - Teams listed in order of most-winning to least-winning.
    - Teams with Most Games Tied:
        - Teams listed in order of most games tied as tiebreaker.
    - Teams with Most Points Scored:
        - Teams listed in order of most points scored as tiebreaker.
    - Check that the teams with Highest Difference of Points Scored to Points Lost
        - Teams listed in order of highest difference as tiebreaker.
    - Teams with Most Games Played:
        - Teams listed in order of most games played as tiebreaker.
    - Teams Ordered by Lexicographic Order of Team Names
        - Check that the teams listed in lexicographic order if all other tiebreakers are equal.
- Check if the leaderboard has listed teams in order of most-successful to least-successful based on game outcomes.

Data Flow:

- Call after addTeam()
- Call after recordGameOutcome()

Test	**Outcome**	**Note**
Print board contains single game	true	Minimal board to print
Print a board before calling recordGameOutcome()	true	Print only the header as output.
Print a board after having successfully calling recordGameOutcome()	true	Board after recording games should be printed
Print a board after recordGameOutcome failed	true	Leader board contains the games before this failure occurs

Questions

How, if at all, would input validation change if you were getting input from a user interface rather than as parameters to methods?

Here, we receive input as parameters in methods, so we only need to check if it is null or empty. However, when obtaining input from users, we must consider various factors, such as ensuring they do not enter integers or Booleans as the team's name. Additionally, users should refrain from entering names in languages other than English. In the recordGameOutcome() function, it is crucial to validate that users do not input scores as floats or decimals.

Explain whether or not boundary cases exist beyond checking the incoming input value.

For incoming input values, we only check if they are in the correct form. During input validation, we ensure they are not null or empty. In boundary cases, on the other hand, we verify that the input string adheres to the constraints of our program. For instance, in the addTeam() method, we ensure that the length of the team name is greater than 0 and less than 15. Additionally, for integers, we perform checks by examining both the maximum and minimum values for that variable.

How does the idea of "control flow" change between opaque (black box) tests and transparent (white box) tests?

In black-box testing, we verify "are we making the right product" by ensuring the correct output for appropriate inputs. In this case, our control flow involves testing the program with various input types and running the program to ensure that each method is executed at least once. In white-box testing, we focus on "are we making the product right" by checking the internal structure of the code. We design test cases to cover every line of code, ensuring that each line runs at least once. This approach helps us identify and address any potential bugs in the code.

Should all permutations of methods result in data flow tests? Explain.

While data flow testing illustrates the sequence in which each function is called and executed, not all permutations of method invocations result in data flow issues. Some permutations that we apply to our program have no impact on the output.

What aspects of test generation (what to test or code to do the testing) do you expect an AI tool to be able to do for you?

AI tools for test generation can assist with a variety of testing tasks, including defining boundary conditions, outlining control flow, and developing test cases for input validation, boundary, control flow, and data flow.

- It can automatically generate test cases based on specifications, requirements, or code under test.

- It can generate both positive and negative test cases to cover a wide range of scenarios and edge cases.

Some AI-powered test automation tools can even generate test cases for us based on our test scenario in plain English. However, it is important to note that AI tools are not a replacement for human testers, and they should be used in conjunction with human expertise to ensure the best possible results