CSCI 3901 Winter 2021

Lab 3: Testing

Name: Dhrumil Rakesh Shah

- **No Team Member (Doing Solo)**

Part 1: Understand the Problem

1) **Ambiguities:**
   a) The "-" column/points lost column: how it is calculated?
      i) **Clarification:** The points lost column is basically the addition of all the points scored by the opponent in all the matches played whether the particular team won or lost.
   b) The leader board string that is created by the createLeaderBoard() method, under that the no. of columns is mentioned for each title: so will that be the maximum length the string titles will store?
      i) **Clarification:** Yes, we can safely assume that each String title of the leader board will hold the maximum length as mentioned in the problem statement prescription. Anything more than that would be considered a boundary test case.
   c) No condition mentioned on the order in which the teams will play against each other. How to avoid collisions?
      i) **Clarification:** Assume that some logic is being used that avoids collisions. No test cases are considered for collisions.
   d) What is the range of points a team can score in a particular match?
      i) **Clarification:** Assuming anywhere from 0 to 20 (one can assume anything) (below some of the test cases are designed on this assumption)

2) **Boundary Conditions:**
   a) Boundary condition for no. of teams playing in the league:
      i) More than 24 teams are added.
      ii) Only 1 team is added to the league.
      iii) 0 teams added to the league
   b) How the leader board be updated if the game is not recorded due to some error faced with the recordGameOutcome() method.
   c) How will the team be added to the league & leader board if we faced an error with addTeam() method.
   d) createLeaderBoard() output problems:
      i) like less columns for any of the titles.
      ii) No spacing between different titles.

3) **Control Flow:**
   a) Start
   b) Loop starts
      i) String Team name as input parameter passed to addTeam(String teanName)
         (1) If (team name is a valid string and it is not in the league and league has less than 24 teams)
            (a) Team name added to league
         (2) Else
            (a) Return false
   c) Loop ends
   d) Leader Board is created by calling createLeaderBoard()
      i) It is an empty leader board with only team names sorted lexicographically, rest all titles are initialized to 0

e) Logic of the matches being played (loop starts)
   i) After every match
      (1) recordGameOutcome(String team1, String team2, int scoreTeam1, int scoreTeam2) is called to record the score of the match played
         (a) If (game was recorded)
            (i) Return true
         (b) Else
            (i) Return false
      (2) Calling to createLeaderBoard() to create an updated leader board
         (a) If (there is a tie between teams)
            (i) If (Using the logic to decide the order of tie-breaking by using below precedence)
               1. Most games won
               2. Most games tied
               3. Most points scored by the team
               4. Highest difference of points scored to points lost
               5. Most games played
               6. Lexicographic order of the team names
         (b) Else
            (i) Normal addition of the data to the table
   ii) Logic to print the leader board string returned by the createLeaderBoard()
f) Loop ends
g) End

4) Normal order in which the methods would be invoked for the problem
   a) **a b i 1 a c d i e i 1 a i 2 b i ii f g ->** as per above mentioned flow

Part 2: Create Test Cases:

1) Input Validation Tests:
   a) **Public boolean addTeam(String teamName)**
      i) Any other data type given as input instead of String and as a String
         (1) Should return false and try again
      ii) Empty string
         (1) Should return false and try again
      iii) Null value passed
         (1) Should return false and try again
      iv) String larger than 15 characters entered
         (assuming from the createLeaderBoard() that the Team Name will use 15 columns)
         (1) Should return false and try again
      v) Only 1 team entered
         (1) Adds the team
      vi) 25th team entered
         (1) Returns false
      vii) Two teams entered
         (1) Adds the teams
      viii) Alphanumeric name entered as team name
         (1) Adds the name (as nowhere it is mentioned that team names shouldn't be alphanumeric
      ix) Returned anything other than true/false/0/1 (boolean)
         (1) Exits the program
      x) Team not added to the league due to some error
         (1) Returns false

- **b) Public boolean recordGameOutcome (Stiring team1, String team2, int scoreTeam1, int scoreTeam2)**
  (not checking for string as we have control over it in the program)
  - **i)** League has only 1/0 teams
    - **(1)** Error while calling the method with just one team
  - **ii)** Any other data type passed to scoreTeam1 & scoreTeam2 instead of int like long/float/double
    - **(1)** Accepts the values as java does implicit type casting
  - **iii)** Negative int values passed to scoreTeam1 & scoreTeam2
    - **(1)** Should return false and does not record the game
  - **iv)** String passed instead of int for scoreTeam1 & scoreTeam2
    - **(1)** Method won't be invoked
  - **v)** 0 passed to scoreTeam1 & scoreTeam2
    - **(1)** Will consider and record the game and go into tie breaking logic
  - **vi)** Returned anything other than true/false/0/1 (boolean)
    - **(1)** Exit the program

- **c) Public String createLeaderBoard():**
  - **i)** Formatting of the string returned not according to the required output
    - **(1)** Output will not be well formatted
  - **ii)** Null league used by the createLeaderBoard()
    - **(1)** Returns a null string
  - **iii)** Less than 15 columns for team name
    - **(1)** The entire team name won't be printed properly/may overlap the other titles
  - **iv)** No data to create the board (league is empty)
    - **(1)** Empty leader board gets generated

2) Boundary Tests:

- **a) Public boolean addTeam(String teamName)**
  - **i)** Empty string passed as team name
  - **ii)** Null value passed as team name
  - **iii)** 1 team name passed
  - **iv)** 25th team name passed

- **b) Public boolean recordGameOutcome (Stiring team1, String team2, int scoreTeam1, int scoreTeam2)**
  (not checking for string as we have control over it in the program)
  - **i)** Negative int passed as scores
  - **ii)** Zero passed as score
  - **iii)** 21 passed as score
    (assumed score to be between 0 to 20)

- **c) Public String createLeaderBoard():**
  - **i)** Negative values when calculating tie breaking by Highest difference of points scored to points lost
  - **ii)** Team name string larger than 15 characters
  - **iii)** All other strings larger than their required length
  - **iv)** The Team title is right aligned instead of left.
  - **v)** Other titles are left aligned instead of right.
  - **vi)** All titles are centre aligned.

3) Control Flow Test:

- **a)** a b i 1 a c d i e i 1 a i 2 a i ii f g

b) a b i 2 a c d i e i 1 b i 2 b i ii f g
c) **a b i 1 a c d i e i 1 a i 2 b i ii f g** (normal flow)
d) a b i 1 a c d i e i 1 b i 2 b i ii f g

4) Data Flow Test:

a) Test1
- **i)** Add team
- **ii)** Empty the league
- **iii)** Create leader board

b) Test2
- **i)** Add team
- **ii)** Create leader board
- **iii)** Record game outcome
- **iv)** Created leader board
- **v)** Add team

c) Test3
- **i)** Create leader board
- **ii)** Add team

d) Test4
- **i)** Add team
- **ii)** Empty the team

e) Test5
- **i)** Add team
- **ii)** Create leader board
- **iii)** record game outcome
- **iv)** create leader board
- **v)** add team

f) Test6
- **i)** Add team
- **ii)** Create leader board
- **iii)** record game outcome
- **iv)** create leader board
- **v)** record game outcome
- **vi)** create leader board
- **vii)** record game outcome (match tied)
- **viii)** create leader board

g) Test7
- **i)** Add team
- **ii)** Record game outcomes
- **iii)** Create leader board
- **iv)** Calling above multiple times

Questions:

- **Getting input from user interface rather than parameters to methods**
  - It would change certain input validation test cases which could be avoided if we internally pass the parameters to the methods.
  - The user could enter wrong data causing us to check each & every data that we accept from user interface
  - Rather passing as parameters, we know what the method expects so we keep that in mind and pass the data, leading to lesser input validation tests.
  - A try-catch block would be required to catch an exception of a very long string or we can use scanner limit for an appropriate number of characters.

    - We even need to catch any NullPointerException caused.

- **Whether or not boundary cases exist beyond checking the incoming input values**
    - Yes, boundary cases exist
        - For the tie-breaking logic the **highest difference of points scored to points lost** could be negative, so we will have to check which is higher value on the negative side
        - There could be only 1 team added to the league
        - No teams added to the league
        - Uninitialized variables within the code, using garbage values
        - There is a tie between teams on every aspect other than lexicographic order

- **How does the idea of "Control flow" change between black box tests & white box tests?**
    - The general idea of control flow is that we go through each & every possible path through the program and try all different combinations throughout the code.
        - So, for white box, we already know the internal working of the program which helps us in eliminating certain redundant an overlapping test cases which saves a lot of time and effort.
        - Sometimes knowing the internal working can also cause on missing out on few of the test cases as we know the working so we might not think of out of the box and illogical tests.
        - And, for black box testing, it's the exact opposite, we don't know the internal working so it causes us to write useless and redundant test cases but the pro is that we test the code from a fresh perspective exploring different options which they might not be explored in white box.

- **Should all permutations of methods result in data flow tests? Explain**
    - I somewhat agree with the above statement, all the permutations majorly cover the entire flow of the program also focusing on the points at which variables receive values and the points at which these values are used within the program.
    - The main idea behind data flow tests is to test the paths in the program/data that force the unexpected uses of data in the program.
    - Manipulating and checking the data variables going in and out of the methods would suffice for data flow tests