CSCI 3901

Software Development Concepts

Faculty of Computer Science

Lab 3: "TESTING"

Kishan Kahodariya B00864907

Dhruv Patel B00868931

Part 1 - Understand the Problem

Clarify any ambiguities that your team may have about the problem

- Does every team that is added to the list, needs to play with every other opponent in the league?
- Can a team can play against same team more than once?
- How is the winner of a match decided? (For example, in football the team with most no. of goals after 90 minutes of play is declared winner whereas in badminton, first player who scores 21 points wins the set.)
- So is the match Time Constrained or Point based?
- Should the team names be case insensitive or not?

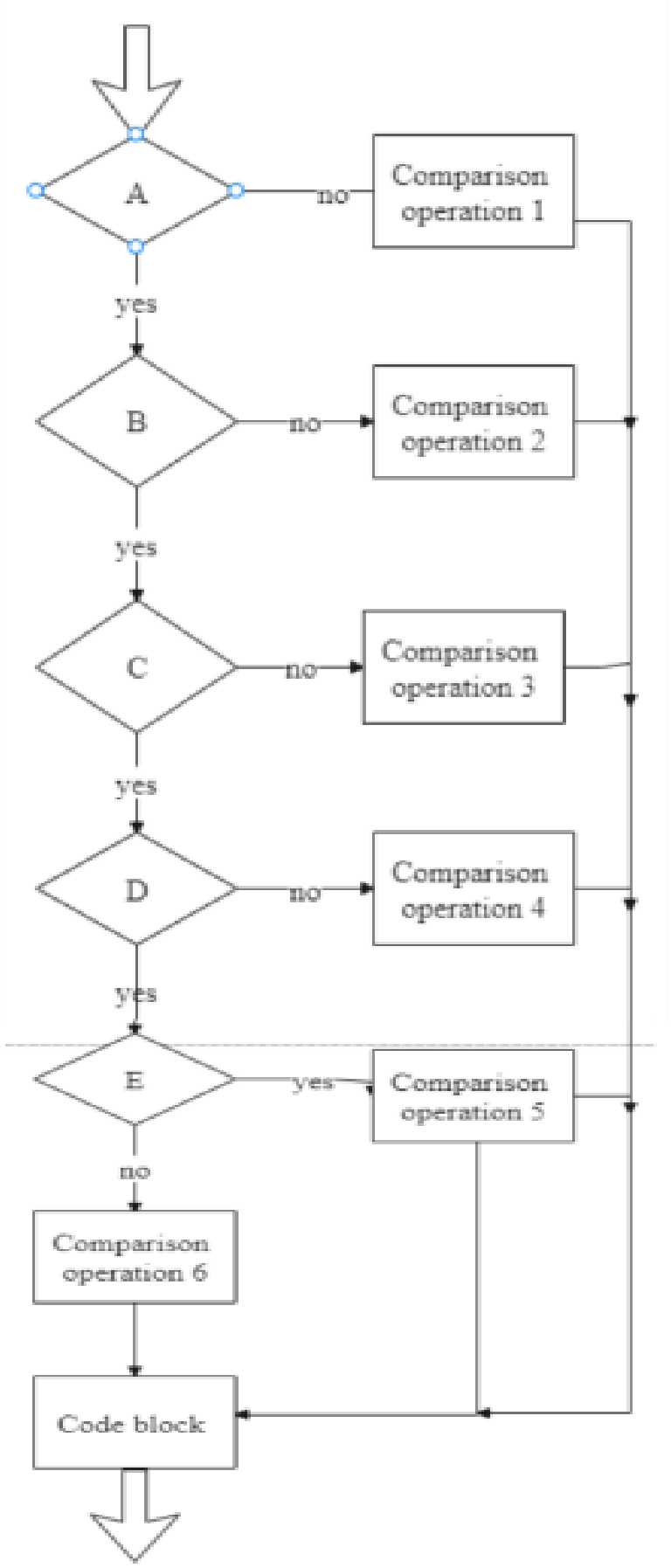
Identify the boundary conditions that exists in the problem statement

- Leader board can add maximum of 24 teams.
- As mentioned in the ambiguities, it's possible for user to add two teams of same name but different in cases (Upper & Lower) and record for both teams will also be same i.e. a duplicate record is generated for same team. (Refer: Test case 5)
- Team name can only have string length of 15 characters (including spaces).
- No. of games won can't exceed double digit (i.e. "00").
- No. of games lost can't exceed double digit (i.e. "00").
- No. of games tied can't exceed double digit (i.e. "00").
- Points scored by each teams ("+") in leaderboard can't exceed 4 digits (i.e. "0000")
- Points lost by each teams ("-") in leaderboard can't exceed 4 digits (i.e. "0000")

Identify the type of control flow

- Every implementation of this program will follow the below mentioned Control Flow tailored for every possible test case:

Conditions	Description
A	Same Games Won
B	Same Games Tied
C	Same Points Scored By The Team
D	Same Difference Of Points Scored To Points Lost
E	Same Games Played
F	Lexicographic Order Of The Team Names


A
no
Comparison operation 1
yes
B
no
Comparison operation 2
yes
C
no
Comparison operation 3
yes
D
no
Comparison operation 4
yes
E
yes
Comparison operation 5
no
Comparison operation 6
Code block

Define the "normal" order in which methods would be invoked for the problem

- In our case, we brainstormed regarding what would be an ideal normal order for which the code will run smoothly without showcasing any exceptions and executing any boundary test cases. We concluded that the best possible scenario in which leaderboard will be generated successfully without any conflict is where every match results in clear winner and loser without any match resulting in tied and also, every team plays equal number of matches.
- The specific order to achieve above mentioned scenario is as follow:

    1. addTeam(String teamname) – Multiple times invocation as needed
    2. recordGameOutcome(String team1, String team2, int scoreTeam1, int scoreTeam2) – Multiple time invocation according to matches played between team 1 and team 2.
    3. createLeaderBoard()

Part 2 – Create test cases

Identify set of input validation tests

Input validation tests:

addTeam(String teamName**):**

Parameter Name	Accepted Datatype	Valid Input	Invalid Input
teamName	String	• String having length between 1 to 15 (including 1 and 15)	• teamName is empty string or null • teamName has more than 15 characters

- Empty string or null value passed
    - Expected outcome : return false

recordGameOutcome(String team1, String team2, int scoreTeam1, int scoreTeam2**):**

Parameter Name	Accepted Datatype	Valid Input	Invalid Input
team1	String	• String having length between 1	• Team1 is empty string or null

		to 15 (including 1 and 15)	• Team1 not present in teamlist
team2	String	• String having length between 1 to 15 (including 1 and 15)	• Team1 is empty string or null • Team1 not present in teamlist
scoreTeam1	Integer	• Positive integer value including 0.	• scoreTeam1 having negative value.
scoreTeam2	Integer	• Positive integer value including 0.	• scoreTeam2 having negative value.

- Empty or null String passed in team1 or team2
  - Expected outcome : return false
- Negative values passed in scoreTeam1 or scoreTeam2
  - Expected outcome : return false

createLeaderBoard()

No input validation test

Identify set of Boundary tests for these boundary cases

Test 1 (Same no. of Wins)

Team	W	L	T	+	-
Anchor	2	0	0	27	19
Salty	1	0	1	20	15
Kitchen Party	1	1	0	11	15
RC	0	2	1	16	23
Hawks	0	3	0	17	20

- In this case, team Anchor is the clear winner with 2 Wins, followed by team Salty with 1 win.
- It is worth noting that team Kitchen Party has also same no. of Wins but has 1 Loss as well, and also team Salty has 1 tied match which means team Kitchen Party stands on 3rd position.
- In case of team RC and Hawks, with both having no wins, the deciding factor was the no. of matches tied by each team.
- Control Flow graph flow: A-B-Comparison Operator 2-Code Block

Test 2 (Same no. of Wins and Tie)

Team	W	L	T	+	-
Salty	2	0	0	32	19
Mavericks	1	1	0	20	15
Warrior	1	2	0	19	15
RC	1	2	0	14	23
Anchor	0	5	1	17	20

- In this case, team Salty and Mavericks are placed in their destined position of 1st and 2nd .
- When it comes to Warrior and RC there are 3 parameters i.e. Win, Loss and Tie which are similar to for every team.
- In such cases, the code is programmed to look at the point scored by each team to determine their respective position in the league table.
- Here, Warrior has score 5 points more than that of RC, so it's place above it.
- Control Flow graph flow: A-B-C-Comparison Operator 3-Code Block

Test 3 (Same no. of Wins, Tie and Point Scored)

Team	W	L	T	+	-	Diff. b/w "+" &" –"
Anchor	4	0	0	27	19	8
RC	3	2	0	18	15	3
Kitchen Party	1	1	0	11	15	4
Salty	1	2	0	34	19	15
Hawks	1	2	1	34	25	9

- Here, first three positions are accurately occupied by team Anchor, RC and Kitchen Party based on no. of Wins registered by each team.
- But for 4th position there is a special case where conditions which proved helped in previous test are of no use here because both teams-Salty, Hawks share same no. of Wins along no. of Loss and tie matches.
- Interestingly, points scored by each is also similar which make it possible for the code to find the best contender for 4th position by comparing the difference between points scored and lost for both teams.
- The position is filled by the team whose difference is greater which in our case is of team Salty.
- Control Flow graph flow : A-B-C-D-Comparison Operator 4-Code Block

Test 4 (based on no. of games played)

Team	W	L	T	+	-	Total Games
Mavericks	3	3	1	27	20	7
RC	3	2	1	27	20	6
Kitchen Party	1	4	1	11	25	6
Salty	1	2	0	12	17	3
Hawks	0	2	0	10	23	2

- In this case, the conflict is the unique one with two teams having similarity in data of more than 2 parameters i.e. no. of win, no. of loss, no. of tie, point scored and point lost.
- This case is one of the boundary cases and in such situation the code should decide the rank of amongst those odd teams by comparing the no. of games played by each team.
- Team which played most no. of games will be ranked on top of other conflicting teams. In this case, Mavericks is placed in 1st position with 7 games compared to 6 games played by RC.
- Control Flow graph flow : A-B-C-D-E-Comparison Operator 6-Code Block

Test 5 (based on lexicographic order of the teams names)

Team	W	L	T	+	-
MAVERICKS	3	3	1	27	20
mavericks	4	2	1	30	20
Kitchen Party	1	4	1	11	25

Salty	1	2	0	12	17
Hawks	0	2	0	10	23

- It is to be noted that the problem doesn't mention whether the teams names are case sensitive or not.
- It is possible for addTeam() method to treat a team with same name but different case i.e. Lower & Upper as two different teams and allow recording of match outcome to the leader board.
- For the 1st & 2nd position, the same team is occupying both position but the only difference is that the 1st position is filled by team MAVERICKS only because it's in upper case despite the fact that it has less no. of win than the team on 2nd position.

- As mentioned in the ambiguities, if we have added a team named "hawks" and we try to add the same team but in uppercase i.e. "HAWKS", code will still permit to do so as there's no clarification on case insensitivity.

- Control Flow graph flow : A-B-C-D-E-Comparison Operator 5-Code Block

Test 6 (based on no. of teams in Leader Board)

- If the no. of teams added to the board is 24 and still, we are trying to add another team ,error message will be displayed as the board only allow maximum 24 teams.

Identify set of Control Flows based on "Normal" order

1. recordGameOutcome → createLeaderBoard → addTeam
   Expected outcome: This should give user message to add the team's first

2. addTeam → createLeaderBoard → recordGameOutcome
   Assumption: if there are no recorded game outcomes then createleaderboard should return null.
   Expected outcome: Leader board should return null.

3. createLeaderBoard → addTeam → recordGameOutcome
   Expected outcome: Leader board should return null.

4. createLeaderBoard → recordGameOutcome → addTeam
   Expected outcome: Leader board should return null.

5. recordGameOutcome → addTeam → createLeaderBoard

Expected outcome: This should give user message to add the teams first

6. addTeam→ recordGameOutcome → createLeaderBoard
   Expected outcome: This should give should give user the correct Leaderboard

Questions

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

- If the method definition remains same then input validation does not change
- For example : addTeam(String xyz) is called after getting input from user then input validation does not change.
- Whereas, if addTeam() is called and inside addTeam we are asking for input from user then we need to include Standard code to get input from user.

Explain whether or not boundary cases exist beyond checking the incoming input values?

- In our view, boundary cases don't exist beyond incoming input values because every time any value is inserted, it's validated and only then added to the leaderboard.
- And if for some reason, the values are declared invalid, it is discarded at that point itself rather than later to avoid inserting false records.

How does the idea of control flow change between black box tests and white box tests?

- In case of Black Box testing we don't worry about how the implementation is being carried out inside each method. We are just concerned with input and output values. On the other hand, white box testing focuses on internal implementation of method.
- For white box tests we make sure that we cover every possible path and includes every statement or path inside the method. For black box testing we don't care if every path or statement is covered; we just test the method with Minimum boundary value, normal value and maximum boundary value and check if the outcome is same as expected outcome in each case.

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

- Yes, in our example we have 3 methods which results in Possible permutation of 6 (3!).
- All possible permutations ae mentioned in the Part-2 section.