# CSCI 3901 Winter 2021

Lab 3: Testing

Due Friday January 29th, 08:30 AST on Brightspace

## Objective

In this lab, you will expand your skills in developing test cases.

**Working alone or in a group of 2**, you will create test cases for a program that creates a leader board for a sports league.

You are not required to develop code for the test cases or to develop test input data for your test cases.

## Preparation

You are going to help manage the leader board for a local sports league. The teams play against one another and the leader board is created by ordering the teams from most winning to least winning. Specifically, the first team on the leader board has the most game wins. If a tie happens between teams then the order of tie-breaking, in order of precedence, is:

- most games won
- most games tied
- most points scored by the team
- highest difference of points scored to points lost
- most games played
- lexicographic order of the team names

You will be writing test cases for a class for this problem. The methods in the class are as follows:

---

`public boolean addTeam(String teamName)` Adds the given team name as one team in the league. Returns true if the team was added to the league. Returns false if the team is already in the league or if any problem arose with the addition.

---

---

`public boolean recordGameOutcome(String team1, String team2, int scoreTeam1, int scoreTeam2)`
Record that `team1` played `team2` and that the ending score was `scoreTeam1` for team 1 and `scoreTeam2` for team 2. Return true if the game was recorded. Return false if the game was not recorded or if any problem arose with the recording.

---

---

`public String createLeaderBoard()` Create a string that is the leader board for the league. The leader board lists all of the teams in the league in the order of most-successful team to least-successful team, based on game outcomes, as described earlier. Each line of the leader board string is as follows:

- team name (use 15 columns)
- number of games won (use 2 columns)
- number of games lost (use 2 columns)
- number of games tied (use 2 columns)
- points scored by the team (use 4 columns)
- points scored against the team (use 4 columns)

Include one space between each column.

There will also be a header row, with titles `Team`, `W`, `L`, `T`, `+`, and `-`, all right justified except "Team".

---

The league has space for a maximum of 24 teams.

Given the following data for `recordGameOutcome` (4 method invocations)

```
"Anchor", "Salty", 15, 12
"RC", "Kitchen Party", 9, 11
"Anchor", "RC", 12, 7
"Salty", "Kitchen Party", 6, 0
```

We get the following leader board

```
Team             W  L  T    +    -
Anchor           2  0  0   27   19
Salty            1  1  0   18   15
Kitchen Party    1  1  0   11   15
RC               0  2  0   16   23
```

## Resources

- None required

## Procedure

### Set-up

1. Ensure that all in your group understand and agree upon the intention of the problem. Do not make assumptions on how someone will implement a program to solve this problem.

### Lab steps

#### Part 1 - Understand the problem

1. Clarify any ambiguities that your team may have about the problem. Record these clarifications.
2. Identify the boundary conditions that exist in the problem statement. Record these boundaries.
3. Outline the type of control flow that *any* implementation for the problem will need to follow, based on the problem statement. Record these flows.
4. Define what would be perceived as the "normal" order in which methods would be invoked for the problem.

#### Part 2 - Create test cases

1. Identify a set of input validation tests for each method as well as the expected outcome for each case.
2. Identify a set of boundary tests for these boundary cases.
3. Identify a set of control flow tests.
4. Identify a set of data flow tests based on your "normal" order.

## Questions

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

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

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

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

## Reporting

1. In one file, list
   - The members of your team.
   - The clarifications and reporting from part 1.
   - The tests from part 2, divided by test type.
   - How complete you think that your test cases are.
   - The answers from the Questions section of the lab.
2. Generate a PDF from the document.
3. Submit the PDF in Brightspace in the Lab/Lab 3 section of the course page in Brightspace.

## Assessment

The assessment will be on a letter grade and will reflect how well you broke down the problem, developed tests, and, most importantly, saw how to take the work done in this lab to broaden it to other contexts (the Questions section).

| Letter grade | Problem clarifications | Extent of the test cases and classification (main weight) | Awareness of the capabilities of current tests | Awareness on using the test structures in broader contexts |
|---|---|---|---|---|
| A (Excellent) | Problem is now clear and good assumptions were made | Considerable non-overlapping tests made for all elements of the program and correctly classified | Demonstrates a good sense for what the tests and testing approaches can do. Knows how variations happen | Clear idea of changes to control and data flows for testing |
| B (Good) | Confusing problem aspects identified. Not all with good conclusions | Considerable tests made for the whole problem, though some overlap, and well classified | Demonstrates a good sense for the testing approaches. May not appreciate the variations | Knows that the control or data flow changes might impact testing, but not spelled out well |
| C (Satisfactory) | Main problem elements identified for clarification | Generally well-tested methods and classification is 75% | Has a reasonable sense of how well the testing is going | Appreciates that control and data flow tests can change, but unsure how |
| D (Marginal Pass) | Just a few problem elements were clarified | A small percentage of tests made for most methods. Unclear if classification helped guide them | Has a rough idea of how well the testing would be, but some of the logic is flawed | Seeing little change in the effects of the flows |
| F (Inadequate) | Lots of ambiguity remains about the problem | Poor testing of the methods and/or no structure to case creation | Little sense on how well testing is happening | Unsure of even the context of the question |