**Dept. of Computer Science & Software Eng., Concordia University**
**COMP 6481 --- Fall 2022**

# Programming and Problem Solving

**Assignment 2 --- Due Sunday, November 27, 2022**

## Part I

**Please read carefully: You must submit the answers to all the questions below. However, this part will not be marked. Nonetheless, failing to submit this part fully will result in you missing 50% of the total mark of the assignment.**

### Question 1

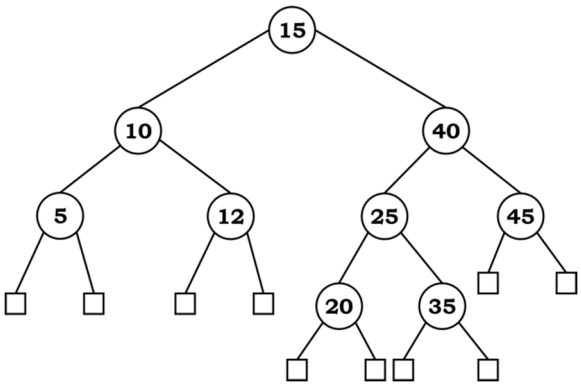
You are given a hash table with size 7 and a hashing function h(k) = k%7. Output the state of the table after inserting the below mentioned values in that specific order when collisions are handled using:

a. Separate chaining
b. Linear probing
c. Double hashing using a second hash function h'(k) = 5 – (k % 5)

The values to be inserted are 19, 26, 13, 48, 17.

### Question 2

Consider the below mentioned AVL tree.




a) Insert key 25 and re-balance the tree if required. Show the progress by drawing each state of the tree and the final state after the insertion.

b) Remove node with value 40 from the original tree and re-balance the tree if required. Show the progress by drawing each state of the tree and the final state after the deletion.

### Question 3

Develop a **well-documented pseudo code** to create a method for open addressing which is like double hashing but instead of using a second hash function use a pseudorandom number generator seeded by the key.

a. Will this be more efficient approach compared to double hashing?
b. Will you prefer this over double hashing?

# Part II

**Purpose:** The purpose of this assignment is to allow you practice Array Lists and Linked Lists, as well as other previous object-oriented concepts.

---

"People throw stones at you and you convert them into milestones."

--- Sachin Tendulkar

Cricket is a sport palyed beween two teams comprising of eleven players each. A game is played on a field with a twenty two yard pitch at the centre of it. This game involves both teams batting and scoring runs. The team scoring more runs in a given number of balls wins the game. The game is adjudicated by two umpiers on field and a third umpire who sits in a monitoring room helps the onfield umpires with the help of technology. There is also a match refree who monitors the conduct of all the players and makes sure the game suns smoothly.

Twenty20 is a special format of the game where each team gets to play twenty overs. A world cup event, organised by cricket's goerning body, the International Cricket Council (ICC), is curently being held in Australia. Twelve teams are participating and are divided into two groups of six teams each. In the first round, each team will play a game aginst all the other teams in their respective group. Top two teams from both groups will move on into the next round to play semi-finals. Qualification of a team for the second round will depend on number of points earned (each victory gives a team two pints, and a tied game gives a team one point) and net run rate in case two or more teams are tied on the basis of points.

As an example, if India has 8 points and are above four other teams, they will move into the next round on points basis. On the contrary if England, Australia and New Zealand all have 7 points each, two teams with a higher run rate will move into the next round.

In this assignment, you will design and implement a tool which will determine if a team will qualify for the second round based on points table for the respective group. You are given few samples, containing information about teams participating in the world cup, and some requests. Each request contains one or more team names. Your program will have to get the input through scanner and must work for inputs other than the sample input as well. You will input this information and will produce an outcome for each team prsent in request(s). The outcome for each request could be one of the below mentioned responses where X represents the team name.

a) Team X qualifies for the second round as it has more points than four other teams.
b) Team X qualifies for the second round as it has a higher net run rate.
c) Team X can't qualify for the second round as it doesn't have enough points.
d) Team X can't qualify for the second round as it doesn't have high enough run rate.

You can assume that all the teams have finished all their games for the first round and all relevant information is made available. The information given to you may not be in any specific order (information structure for each team will be same but order in which teams are entered will not be same). A sample of team information input data is depicted in Figure 1 (TeamName, number of matches played, number of matches won, number of matches lost, net runrate, and number of points), and a sample of request data is depicted in Figure 2. A detailed description of all the details that you have to implement for this assignment is made avaialble after these two figures.

```
GROUP A
ENGLAND         5 4 1 +2.464 8
AUSTRALIA       5 4 1 +1.216 8
SOUTH_AFRICA    5 4 1 +0.739 8
SRI_LANKA       5 2 3 -0.269 4
WEST_INDIES     5 1 4 -1.641 2
BANGLADESH      5 0 5 -2.383 0

GROUP B
PAKISTAN        5 5 0 +1.583 10
NEW_ZEALAND     5 4 1 +1.162 8
INDIA           5 3 2 +1.747 6
AFGHANISTAN     5 2 3 +1.053 4
NAMIBIA         5 1 4 -1.890 2
SCOTLAND        5 0 5 -3.543 0
```

Figure 1. Illustration of team information input

```
AUSTRALIA
NEW_ZEALAND
NAMIBIA
SOUTH_AFRICA
```

Figure 2. Illustration of Requests

**I)** Create an interface named **Groupable** which has a single parameter boolean method called isInTheGroup (Team T) where T is an object of type **Team** described below.

**II)** The **Team** class, which must implement the **Groupable** interface, has the following attributes: a teamID (String type), a teamName (String type), a gamesPlayed (int type), a gamesWon (int type), a games lost (int type), a netRunRate (double type), a points (int type). It is assumed that team name is always recorded as a single word (_ is used to combine multiple words). It is also assumed that no two teams can have same teamID. You are required to write implementation of the **Team** class. Besides the usual mutator and accessor methods (*i.e.* getTeamID(), setTeamName(), *etc.*) class must also have the following:

a) Parameterized constructor that accepts seven values and initializes the teamID, teamName, gamesPlayed, gamesWon, games lost, netRunRate, and points to these passed values;

b) Copy constructor, that takes in two parameters, a Team object and a String value. The newly created object will be assigned all the attributes of the passed object, with the exception of the teamID. teamID is assigned value passed as the second parameter to the constructor. It is always assumed that this value will correspond to the unique teamID rule;

c) clone() method, that will prompt a user to enter a new teamID, then creates and returns a clone of the calling object with the exception of the teamID, which is assigned the value entered by the user;

d) Additionally, class should have a toString() and an equals() methods. Two teams are equal if they have the same attributes, with the exception of the teamID, which could be different.

e) This class needs to implement interface from part I. The method isInTheGroup that takes in another Team object T and should return true if T is from the same group as the current team object, or vise versa; otherwise it returns false.

**III)** The **TeamList** class has the following:

(a) An inner class called **TeamNode**. This class has the following:

i. Two private attributes: a team object and a pointer to a TeamNode object.
ii. A default constructor, which assigns both attributes to null.
iii. A parameterized constructor that accepts two parameters, a Team object and a TeamNode object, then initializes the attributes accordingly.
iv. A copy constructor.
v. A clone () method
vi. Other mutator and accessor methods.

(b) A private attribute called head, which points to the first node in this TeamList.

(c) A private attribute called size, which always indicates the current size of the list (how many nodes are in the list).

(d) A default constructor, which creates an empty list.

(e) A copy constructor, which accepts a **TeamList** object and creates a copy of it.

(f) A method called addToStart(), which accepts one parameter, an object from Team class, creates a node with that passed object, and inserts this node at the head of the list.

(g) A method called insertAtIndex(), which accepts two parameters, an object from the Team class, and an integer representing an index. If the index is not valid (a valid index must have a value between zero and size-1), then the method must throw a **NoSuchElementException** and terminates the program. If index is valid, then the method creates a node with passed Team object and inserts this node at the given index. The method must properly handle all special cases.

(h) A method called deleteFromIndex(), which accepts one int parameter representing an index. If index is not valid, method must throw a **NoSuchElementException** and terminate the program. Otherwise, node pointed by that index is deleted from the list. The method must properly handle all special cases.

(i) A method called deleteFromStart(), which deletes the first node in the list (the one pointed by head). All special cases must be properly handled.

(j) A method called replaceAtIndex(), which accepts two parameters, an object from Team class, and an integer representing an index. If index is not valid, the method simply returns; otherwise, object in list at passed index must be replaced with the object passed.

(k) A method called find(), which accepts one parameter of type String representing a teamID. Method then searches the list for a teamNode with that teamID. If such an object is found, then method returns a pointer to that teamNode; otherwise, method returns null. The method must keep track of how many iterations were made before the search finally finds the team or concludes that it is not in the list.

(l) A method called contains (), which accepts a parameter of type String representing a teamID. Method returns true if a team with that teamID is in the list; otherwise, the method returns false.

(m) A method called equals (), which accepts one parameter of type TeamList. Method returns true if the two lists contain similar teams; otherwise, the method returns false. Recall that two Team objects are equal if they have the same values except for the teamID, which can, and is expected to be, different.

Finally, here are some general rules that you must consider while implementing above methods:

- Whenever a node is added or deleted, the list size must be adjusted accordingly.
- All special cases must be handled, whether the method description explicitly states that or not.
- All clone() and copy constructors must perform a deep copy; no shallow copies are allowed.
- If any of your methods allows a privacy leak, you must clearly place a comment at the beginning of the method 1) indicating that this method may result in a privacy leak 2) explaining reason behind the privacy leak. Please keep in mind that you are not required to implement these proposals.

**IV)** Now, you are required to write a public class called **TournamentResults**. In main() method, you must do the following:

(a) Create at least two empty lists from TeamList class (needed for copy constructor III (e)).

(b) Input team information as depicted in Figure 1 to initialize one of the TeamList objects you created above. You can use the addToStart() method to insert the TeamNode objects into the list. However, the list should not have any duplicate records, so if the input has duplicate entries, your code must handle this case so that each record is inserted in the list only once.

(c) Input request information as depicted in figure 2 and create **ArrayList** from the contents then iterate through each of the teams. Process each of the teams and print the outcome whether the team will move to the next round or not. A sample output for a given input is mentioned below. Again, your program must ask for the input information through console as your program will be tested against similar input information.

(d) Prompt the user to enter a few teamIDs and search the list that you created from the input for these values. Make sure to display number of iterations performed.

(e) Following that, you must create enough objects to test each of the constructors/ methods of your classes. The details of this part are left as open to you. You can do whatever you wish if your methods are being tested including some of the special cases. You must also test the isInTheGroup () method.

```
AUSTRALIA qualifies for the second round as it has higher run rate.
NEW_ZEALAND qualifies for the second round as it has more points than four other teams.
NAMIBIA can't qualify for the second round as it doesn't have enough points.
SOUTH_AFRICA can't qualify for the second round as it doesn't have high enough run rate.
```

Figure 3. A sample outcome of the enrolment request

Some general information:

a. Do not use any external libraries or existing software to produce what is needed; that will directly result in a 0 mark!

b. Again, your program must work for any input data. The sample input provided with this assignment is only a possible version, and must not be considered as the general case when writing your code.

**General Guidelines When Writing Programs:**

- Include the following comments at the top of your source codes.
  // Assignment (include number)
  // Question: (include question/part number, if applicable)
  // Written by: (include your name and student id)
  // ----------
- In a comment, provide a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.
- Display a welcome message which includes your name(s).
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output must be displayed with clear messages and in an easy-to-read format.
- End your program with a closing message so that the user knows that the program has terminated.

**JavaDoc Documentation:**

Documentation for your program must be written in **javaDoc**.
In addition, the following information must appear at the top of each file:

| | |
|---|---|
| Name and ID | (include full name and ID) |
| Assignment # | (include the assignment number) |
| Due Date | (include the due date for this assignment) |

**SUBMISSION INSTRUCTIONS**

**Submission format:** All assignment-related submissions must be adequately archived in a ZIP file using your ID(s) and last name(s) as file name. The submission itself must contain your name(s) and student ID(s). Use "official" name only – no abbreviations/nick names; capitalize the "last" name. Inappropriate submissions will be heavily penalized. **IMPORTANT:** For Part II of the assignment, a demo for about 5 to 10 minutes will take place with the marker. You **must** attend the demo and be able to explain their program to the marker. The demo schedule will be determined and announced by the markers, and students must reserve a time slot for the demo.

**Now, please read very carefully:**

- **If you fail to demo, a zero mark is assigned regardless of your submission.**
- **If you book a demo time, and do not show up, for whatever reason, you will be allowed to reschedule a second demo but a penalty of 50% will be applied.**
- **Failing to demo at second appointment will result in zero marks and no more chances will be given under any conditions.**

**EVALUATION CRITERIA**

**IMPORTANT: Part I** must fully be submitted. Failure to submit that part will cost 50% of the total marks of the assignment!

| **Total** | **10 pts** |
|---|---|
| **Documentations** | **1 pt** |
| **JavaDoc** documentations | 1 pt |
| **Tasks** | **9 pts** |
| Task#1 | 1 pts |
| Task#2 | 2 pt |
| Task#3 | 3 pts |
| Task#4 | 3 pts |