CS210 with Dr. Mohammed Akour **Term Project 1** **Weight 10%** **Due: End of Thursday October 14, 2021**

The Qiddiya SIM Tool

Qiddiya, the new entertainment city is situated 45km from the Saudi capital of Riyadh. Also flagged as the "Kingdom's capital of entertainment", there are more than 45 projects, where visitors will have access to over 300 recreational and educational **facilities** designed around five cornerstones of development that drive the strategy: Parks & Attractions, Sports & Wellness, Motion & Mobility, Nature & Environment, and Arts & Culture. The development will also be home to Six Flags Qiddiya, an extension of the American theme park, with six themed lands, which includes the world's longest, tallest and fastest roller coaster and the world's tallest drop-tower ride. These cornerstones will include a Formula One-standard racing track, a 20,000-seat cliff-top stadium, an 18,000-seat indoor arena, an aquatic center, and a sports hub, as well as a 2,000-seat performing arts theatre and a cinema.

By 2030, Qiddiya hopes to draw 17m visitors annually and build new sectors that will contribute up to 17bn riyals ($4.5bn) to Saudi Arabia's gross domestic product, employing some 25,000 people. [Source: https://qiddiya.com]

Your student team is employed to build a simulation tool to facilitate the booking of the Qiddiya facilities by visitors of all ages. Visitors can book various facilities (1 to 7) for a particular date, which can be a weekday or a weekend, with varying prices for the tickets. Visitors also get special discounts on joyous occasions such as the Eid holidays and National day celebrations.

F_ID	Facility	Weekdays Price	Weekend price	Special Discount
1	Six Flags	200	250	20%
2	F1	100	150	20%
3	Stadium	30	50	50%
4	Arena	30	50	50%
5	Aquatic Center	50	100	25%
6	Theatre	20	50	10%
7	Cinema	20	50	10%

Table 1. Ticket prices for various facilities at Qiddiya

Your sim tool reads data as sets of commands, in batches, processes this data and computes the cost of tickets. This tool would use a simply linked list to implement the various attributes, methods and classes written in the java programming language.

Your project implements three classes namely Node, List and Solution.

The following provides an API for your project.

Node	Description

int ticketID;
int vID;
int vDate;
int FID;
String vName;
Node next;
	// A unique ticket id;
	// Visitor ID number;
	// Visit date DDMMYYYY;
	// Facility ID, takes values between 1 and 7 (inclusive)
	// Name of the Visitor
	// Node next

Node()
Node(,,,)
 	
String toString()
	// default constructor
	//override constructor as necessary. You can edit the params as needed.
	//returns a String with the contents of the Node as follows:
	ticketID vID vDate FID vName

The following is the API for the List class. This implements a Singly Linked List consisting of Nodes from the Node class.

List	Description
Node Head; int size;	// Anchors the Head of the List // Maintains the size of the List

List();
 	
insert(Node N)
 	
void remove(vID)
 	
int cost(vID)
 	
void print1(vID)
void print2(vDate)
void print3(FID)
void print4(ticketID)
	// default constructor, you may have overloaded constructors
	
	// inserts the node N in the correct position, ordered by the date. e.g. date1 is 14022021; date2 is 01032021. date1 appears before date2. Assume all months have exactly 30 days. There are 12 months in a year. If dates are equal (we assume inserting at the beginning).
	
	We also assume that
	• The 15th of any month receives the special discount (weekday price)
	• The 7, 14, 21 and 28 are weekends.
	• All other days are weekdays.
	
	//removes the first occurrence of a node with matching vID.
	
	//This method searches for all Nodes with vID and computes the cost of ALL visits for a visitor with vID. The return value is rounded off (int rounding)
	
	//searches for vID in the list. Prints ALL nodes with vID.
	//searches for vDate in the list. Prints ALL nodes with vDate.
	//searches for FID in the list. Prints ALL nodes with FID.
	//searches for ticketID in the list. Prints ONE node with ticketID.
	
	NOTE: print methods call Node.toString() as necessary.

The following is the API for the Solution class. This class implements the main method, makes appropriate data Input/Output calls and implements data structures and all necessary method calls.

Solution	Description
//Attributes NA	Not applicable
main()	Implements the main method; reads data from console, processes information and make appropriate calls as necessary.

Sample Input

Here is a sample input

```
1 1225 101 10092021 3 Ahmed Baloshi
1 1226 101 10092021 5 Ahmed Baloshi
1 1227 101 10092021 7 Ahmed Baloshi
3 101
2 101
4 101
```

Sample Output

For the above sample input, the following would be printed on the console.

```
148
1226 101 14092021 5 Ahmed Baloshi
1227 101 15092021 7 Ahmed Baloshi
```

Explanation

1 1225 101 10092021 3 Ahmed Baloshi 1 1226 101 14092021 5 Ahmed Baloshi 1 1227 101 15092021 7 Ahmed Baloshi
Your program reads a set of commands from the console using the following format: <command> <ticketID> <vID> <vDate> <FID> <vName> The line starting with integer 1 indicates that insert call would be made. No Output is generated for the insert call.
3 101
The line starting with integer 3 indicates that cost method would be called. The program computes the cost for all visits by Visitor with vID = 101. In the list, there are 3 nodes with vID = 101. Node with ticketID=1225 has a visit cost = 30 (weekday) Node with ticketID=1226 has a visit cost = 100 (weekend) Node with ticketID=1227 has a visit cost = 18 (special discount); 10% discount on 20 is 18 Hence total cost = 30+100+18=148 148 will be displayed on the console
2 101
The line starting with integer 2 indicates that the first node with vID=101 will be removed from the list. No output is generated. This removes the node that contains the following information 1225 101 10092021 3 Ahmed Baloshi
4 101
The line starting with integer 4 indicates that All visitor details for vID=101 will be printed on console separate by new line character at the end of each line. 1226 101 14092021 5 Ahmed Baloshi 1227 101 15092021 7 Ahmed Baloshi

Similar to the above, your program implements the following commands:

Legend for the acceptable commands: 1. insert details for a new node 2. removes node from list 3. computes the Cost of ALL visits for this vID and displays the total. 4. prints All nodes with vID. 5. prints ALL nodes with vDate. 6. prints ALL nodes with FID. NOTE: If no output is generated, your program prints 0 on the console.

Evaluation:

You are allowed to work as a group with maximum 2 to 3 members in a group. Your work's evaluation would be based on Code inspection and successful execution of *k* number of test cases. The instructor reserves the right to determine the scores of each test case.

Test-cases will be posted on hackerrank, students will have unlimited number of opportunities to post and test their project until the due date. The system will not take any submissions after the due date.

Code Inspection:

The code would be inspected by the instructor. The instructor would determine the score to be given for code inspection. Generally, a readable code (indentation, clear scope definition) is required. For this project, there are no limitations on time and memory usage. Hackerrank checks for plagiarism. If the similarity of your submission is more than 50%, you will be awarded a ZERO in the project.

Submission:

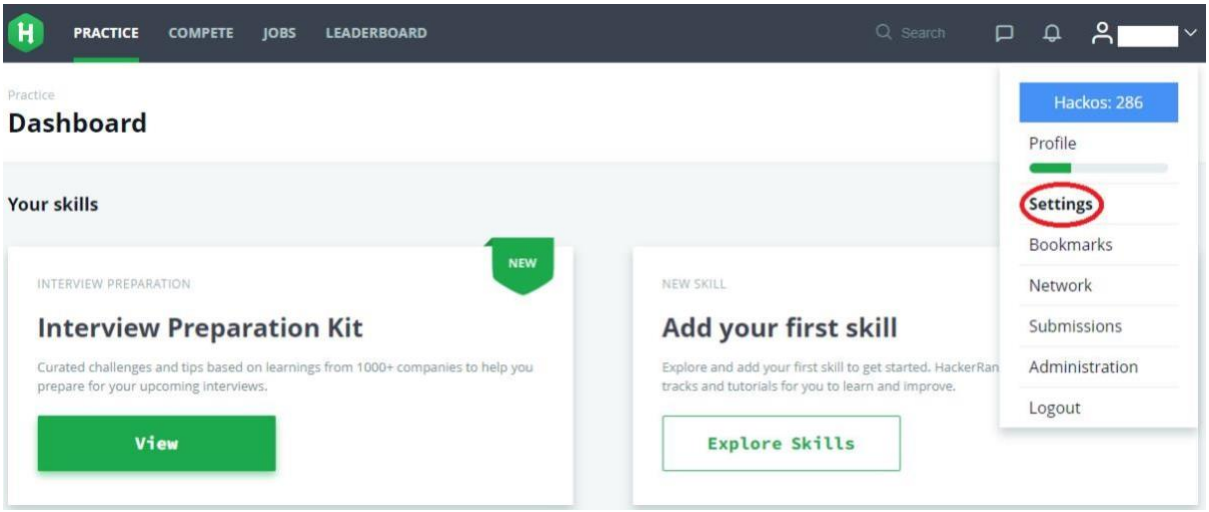
Submission on Hacker Rank

Step 1.

Register on https://www.hackerrank.com/

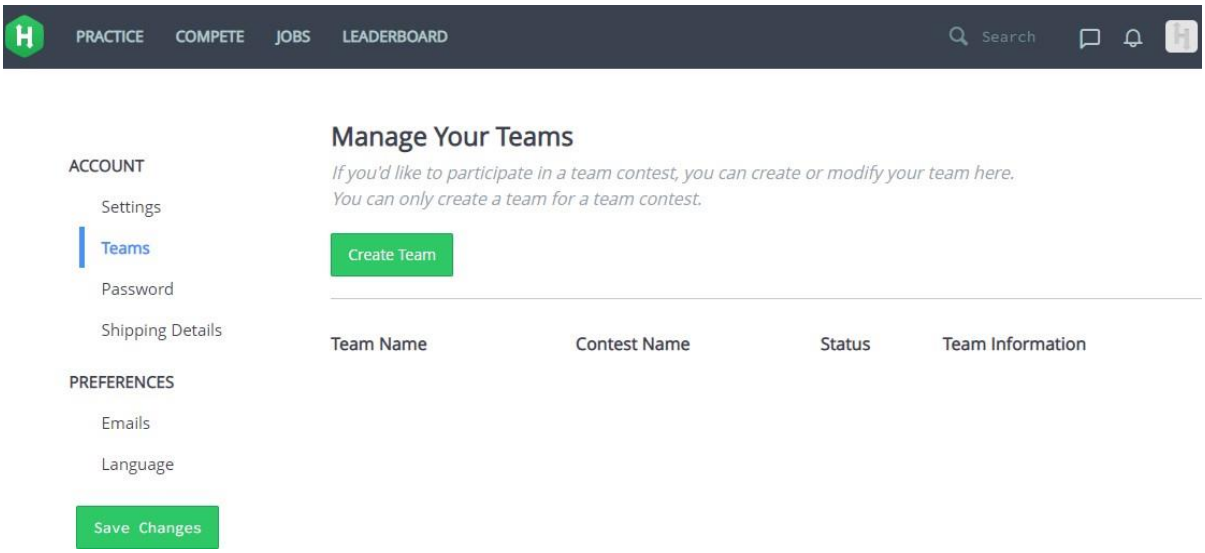
Step 2.

Go to settings



Step 3.

Create a Team. For this project you may work alone (1 person per team) or up to 2 persons per team.



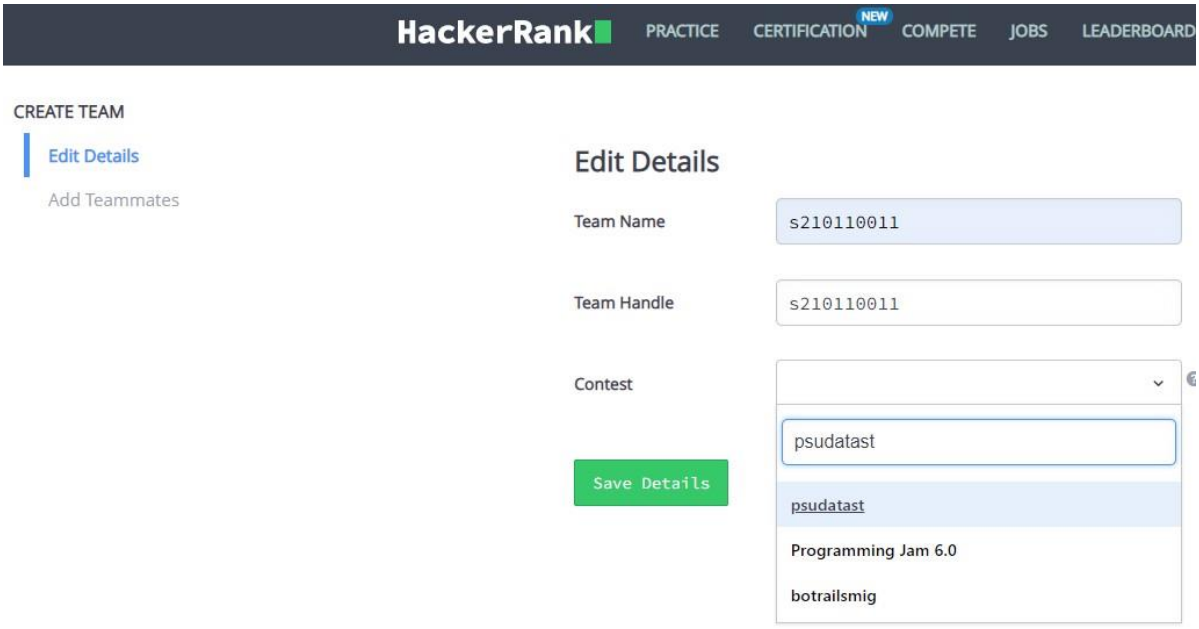
Step 4.

Edit Team details.

Important NOTE:

1. Team name must of "S" your Student ID. If it is 2 students per team, S210110123S210110124

2. Join contest "psudatast"



Step 5:

Go to https://www.hackerrank.com/psudatast

Start working on your project.

Submission Dead-Line:

The submission deadline is final. Late Submissions will be awarded ZERO points.

Plagiarism:

Be warned, all code submitted would be compared using hackerrank. Any similar code would result in ZERO earned for both group members as well as all other groups with similar code. NO EXCEPTIONS!

Important Notes:

- It is the student's responsibility to check/test/verify/debug the code before submission.
- It is the student's responsibility to check/test/verify all submitted work (including jar files)
- It is the student's responsibility to verify that all files have been uploaded to the LMS.
- For each project, instructor will provide 1-2 sample test-cases to verify the execution of your program.
- After an assignment/project has been graded, re-submission with an intention to improve an assignments scores will not be allowed.
- After the assignment/project has been graded, the instructor will post test-cases used for grading on the website.
- The Instructor has the right to share project execution reports that may have been auto-generated on the course website.