

# Modern University for Business and Science

## School of Computer and Applied Sciences

### CSC325 – Python Programming Assignment 1 2019-2020

Student's Name: \_\_\_\_\_

(First)

(Father)

(Family)

Student ID Number: \_\_\_\_\_ Instructor's Name: \_\_\_\_\_

Assessment weight: 40% of the course

#### LEARNING OUTCOMES

This exam will cover Learning Outcomes:

1. Introduction to program flow control,
2. The basics of Python, and built-in data types: Lists, ranges, tuples, dictionaries and sets.
3. I/O in Python
4. Conditional statements and looping
5. Functions, function arguments, and control flow
6. Classes and Inheritance
7. File operations using Python

#### DEPARTMENTAL USE ONLY

Score 100

**Kindly, make sure that you read and understand the following points:**

- The **PLAIGAIRISM** penalty will be an “F” on the assignment.
- each student has to submit his own project/assignment. **Group projects are not allowed.**
- Your name should written onto these sheets; moreover, your answers should be applied on a new document/file to be submitted, along with this file, to the Instructor.
- Total number of pages is **3 pages**.
- Submission Deadline: **May 24, 2020 at 11pm**

**DO YOUR BEST!**






## Champions League Final

The organizers of the football Champions League want to computerize the information of the final match including the squad of the two teams and the score sheet.

For reasons of simplicity, we assume that the number of the players on the field (lineup) of a team is **4** (instead of 11). Each team has also a set of players on the bench (substitutes) for eventual substitutions during the match.

Write the python code that defines the following classes.

- A. Create a class **Member** that contains the following: (7.5 pt)
1. Fields representing the name, the age and the nationality.
  2. A constructor to initialize the fields.
- B. Create a class **Coach** that inherits from the class **Member** and has also the following: (7.5 pt)
1. A field representing the number of years of experience in the domain of coaching.
  2. A constructor to initialize the fields.
- C. Create a class **Player** that inherits from the class **Member** and has also the following: (7.5 pt)
1. Fields representing the jersey number, the position (ex: goalkeeper, defender, midfielder or forward) and a boolean indicating whether the player is the captain.
  2. A constructor to initialize the fields.
- D. Create a class **Referee** that inherits from the class **Member** and has also the following: (7.5 pt)
1. A field representing the number of matches he controlled so far.
  2. A constructor to initialize the fields
- E. Create a class **Team** that has the following:
1. The name.
  2. The nationality
  3. The coach.
  4. The lineup  ch is a list of **4** players.
  5. The bench of substitutes which is a **List** of players. (5 pt)
  6. A constructor to initialize the name, the nationality and the coach. (5 pt) 
  7. A function **assignLineup(PlayerGK, PlayerDefense, PlayerMiddle, PlayerForward)** that assigns the lineup by the players passed as parameters. (5 pt)
  8. A function **addSubstitute(P)** that adds a player **P** to the bench. (5 pt)
  9. A function **substitution(inP, outP)** that substitutes the lineup player whose jersey number is **outP** by the bench player whose jersey number is **inP**. Note that the outgoing player takes place on the bench. Also, if the outgoing player is the captain then the incoming player becomes the captain. (5 pt) 
  10. A function **SaveLineup()** that allows to store the list of players in a text file in the format of the following example. The symbol (C) means that the player is the captain: (10 pt)

Team: Juventus		Coach: Sarri
Jersey Number	Name	Position
77	Buffon	Goalkeeper
3	Chiellini	Defender (C)
10	Dybala	Midfielder
7	Ronaldo	Forward

F. Create a class **Match** that contains the following:

1. The event (ex: Champions League Final 2020).
2. The stadium name.
3. The referee.
4. The home team.
5. The away team.
6. The home team scorers which is a **List**.
7. The away team scorers which is a **List**. (5 pt)
8. A constructor to initialize the fields. Of course the match starts goalless! Also, the number of controlled matches of the referee is incremented after each match. (5 pt)
9. A function **addGoal(teamName, playerName, minute)** that adds a goal to the corresponding **List** according to the team name passed as parameter. For each goal, we add a string composed from the name of the scorer followed by the minute when he scores the goal (ex: "Ronaldo 52"). (5 pt)
10. A function **displayScore()** that displays the score of the match in the following format: NbGoalsHomeTeam : NbGoalsAwayTeam (ex: 1:3) (5 pt)
11. A function **displayWinner()** that displays, according to the number of goals of each team, a message in the following format:  
(WinnerTeamName) "are the champions of" (Event). If the numbers of goals of both teams are equal, than display "Draw! Go to penalties". (5 pt)
12. A function **PrintScoreSheet()** that store in a csv file the details of the match in the format of the following example: (10 pt)

Event: UCL Final 2020	Stadium: Ataturk Stadium	Referee: Brych (Germany, 39 years)
-----	-----	-----
-	-	--
Home Team	VS	Away Team
Juventus (Italy)	2 : 1	Real Madrid (Spain)
(Ronaldo 25) (Dybala 60)		(Benzema 13)
Juventus are the champions of UCL Final 2020		