Object-Oriented Programming *COURSE & LAB* – Fall 2021
(BS-CS-F20 Morning & Afternoon)

Project # 1

Assigned on: ***Monday, January 03, 2022***

Submission Deadline: ***Monday, January 10, 2022 (till 5:00 PM)***

Instructions:

- **This is group project of 1- 3 members.**
- This project will be counted towards your marks in both **OOP Course and Lab**.
- You are required to create a **multi-file project** for this project.
- Do **NOT** copy even a single line of code from any other person or book or Internet or any other source.
- This project needs to be submitted in **Soft** form. The **Submission Procedure** is given below.

Submission Procedure:

i. Create an empty folder. Put all of the ***.CPP* and .H file(s)** of your assignment in this folder. Do NOT include any other files in your submission, otherwise your submission will NOT be graded. Don't forget to mention your Name, Roll Number and Section in comments at the top of each CPP file.

ii. Now, compress the folder (that you created in previous step) in **.RAR** or **.ZIP** format. The name of the RAR or ZIP file should be ***exactly*** according to the format:

`Mor-A1-BCSF20M123` (for Morning section) OR

`Aft-A1-BCSF20A456` (for Afternoon section),

where the text shown in **BLUE** should be your **complete Roll Number**.

iii. Finally, submit the (single) **.RAR** or **.ZIP** file through **Google Classroom**.

iv. There will be a short presentation and viva of the project and its date will be announced later.

Note: *If you do not follow the above submission procedure, your project will NOT be graded and you will be awarded ZERO marks.*

These *good programming practices* will also have their (significant) weightage in the marking of this assignment:

- Comment your code intelligently. **Uncommented code will NOT be given any credit.**
- Indent your code properly.
- Use meaningful variable and function names. Use the **camelCase** notation.
- Use meaningful prompt lines/labels for input/output.

If your program **does NOT compile correctly** or it **generates any kind of run-time error** (dangling pointer, memory leak etc.) you will get **ZERO marks** in the whole project.

Cricket game

You need to design the game using classes and objects concept. Apply the relevant OOP concepts that you studied yet wherever they are applicable. You have to design at least two classes (Team and Player) with at least the below mentioned attributes and functions. You may define additional attributes and functions to ease and support your implementation. You need to think and adjust each attribute and function to its appropriate and relevant class. You may have an optional driver class (to invoke appropriate class members). Driver class basically creates relevant objects and calls relevant member functions..e.g. bowling() and batting(). You may skip designing this class and put driver class logic in main().

You need to implement following functionality in background.

- Bowling and batting must be continuously being called till one of the team wins the match (or in case of match draw)
- Whenever a bowling function is called, this will decrement the number of balls left.
- Whenever a batting function is called, this might (or might not) increment the score.
- Whenever a batsman is declared as ''out'', number of wickets of relevant team gets decremented
- In case of noball, update the attributes: +1 score, +1 ball
- In case of wide, +1 score

Some obvious Attributes: TeamID, Number of wickets, balls, playerID, score, statusWin

Some obvious Functions: DecisionOfGame() DecisionOfToss() Batting() Bowling()

1. Welcoming screen with menu

The game starts with a Welcoming Menu screen with at least three options:

```
**Welcome**
Select an option (1-3):
    1. Play
    2. Rules
    3. Exit
```

On entering 1, user is directed to cricket game. Option 2 displays the rules of the game. Option 3 exits the program.

2. Players and team selection

First of all, the user selects a team for tournament. Display a list of 5 teams (as shown below) and allow user to select one of the team from the list.

```
Select your team from the following.
    1. Pakistan
    2. Australia
    3. New Zealand
    4. Zimbabwe
    5. Sri Lanka
Your team: 5
```

Once the user has selected the team, allow the user to select squad of 11 players. Display the list of at least 16 players from which the user will select 11 players. You need to read the respective file (as per the team selected by the user) and display the list of players by reading from that file.

Select your squad of 11 players from the following.

1. [Player name of the respective team; as read from the file]
2. [Player name of the respective team]
   …

16. [Player name of the respective team]

Enter the number to select the relevant player.
Player 1: 1
Player 2: 3
Player 3: 11
…
Player 11: 16

Maintain a text file named **teamPlayers.txt** that consisting of a list of 16 players in each team. The file contains the name of players in each line. Arrange the record of players in the following order:

1. Pakistan
2. Australia
3. New Zealand
4. Zimbabwe
5. Sri Lanka

That is, first 16 lines refer to the players of Pakistan team, the next subsequent 16 players are of Australian team and so on. Use file random access methods (seekg/seekp) to jump to the respective team for fetching the record of players of a specific team.

3. Batting order modification

After selecting team players in step 2, user is asked if he/she wants to change the batting order. The user is asked about two things: 1) the id number of the players which needs to be modified and 2) Position number in the team to insert that player.

Here is your selected team squad:

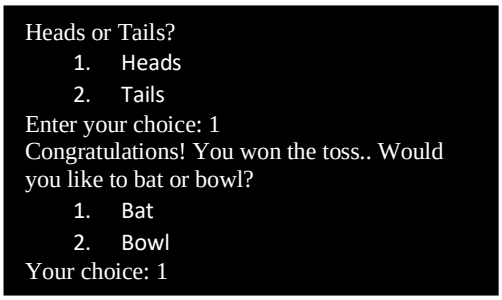
1. [Display the player number at ID 1]
2. ..
3. ..

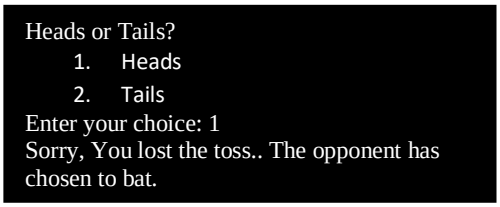
11. …

Do you want to modify the batting order (Y/N)? Y
Please enter the id number of the players which needs to be modified: 4
Please specify the position number in the team to insert the player: 1
[Display the modified list]

4. Toss

The next step is toss. If a user wins the toss then he is allowed to choose if he wants to bowl/ bat. The toss functionality is implemented using the concept of random numbers generation.





5. Batsman behavior

There can be three categories of players: 1) Bowler, 2) Batsman, and 3) Allrounder. Implement all the necessary relationships that we studied till now.

To simulate **batsman behavior**, you will need to have one attribute: isStrike(): true or false. True value indicates that the batsman has played the ball whereas false indicates that the batsman has acted defensive and skipped playing that ball.

---

6. Balls and bowling behavior

Following four categories of balls must be considered:

- Noball
- Wide
- Bouncer
- Spin

To simulate the **behavior of bowling**, you need to keep following attributes (a constant defined attribute and user entered attribute must be compared to label the category of ball). The attributes starting with const_ must be defined as a constant. You may hardcode any dummy value in it. For example,

const_distanceV = 50 feet

const_distanceV: that tells how long is the pitch.

Entered_distanceV: value to be taken as an input from the user. This has to be compared with Const_distanceV to label the ball action as noball.

Const_distanceH: that tells the valid width allowed for bowling. (e.g. if the Entered_distanceH becomes larger than this value then it will result in a wide ball)

entered_distanceH: value to be taken as input from the user. This has to be compared with Const_distanceH to label the ball action as wide.

Const_height: maximum height allowed, above which the bowling will be considered as a bouncer.

Entered_height: value to be taken as input from the user. This has to be compared with Const_distanceV to label the ball action as a bouncer.

Entered_angle_of_deviation: value to be taken as input from the user. It tells the angle of deviation to decide whether the ball is a spin or not. Deviation of angle 5 degrees is allowed from 90 degrees. If deviation crosses +5 or -5 limit then this will be categorized as a spin.

Assuming the user won the toss and chose to bowl

Labelling rules: (No randomization allowed while labelling-compare user input with constant values and label)

Batsman behavior (Use randomization here, randomly assign value to isStrike flag)

Note: If the user chose to bat, then you have to perform reverse. You need to randomly assign values to Entered_distanceV and other relevant attributes. The user will now tell whether to strike or not and you need to assign value to isStrike attribute using user input.

		Batsman behavior	
Ball label	**Criteria**	**If batsman strikes**	**If batsman does not strikes**
Noball	If Entered_distanceV < Const_distanceV	Choose randomly from 4, 6	0 runs
Wide	***add appropriate rule***	Always make 4 runs	0 runs
Bouncer	***add appropriate rule***	Always make 6 runs	0 runs
Spin	***add appropriate rule***	Choose randomly from 0,1,2,4, 6	0 runs

Once bowling label has been found, following categories of decisions must be considered:

- HitWicket
- LBW
- Catchout

Decision	**Bowling label**
HitWicket	Bouncer, spin, normal
LBW	spin, normal
Catchout	Bouncer, spin, normal

☺ GOOD LUCK! ☺

Remember: Honesty always gives fruit (no matter how frightening is the consequence); and Dishonesty is always harmful (no matter how helping it may seem in a certain situation)