UPES

UNIVERSITY OF TOMORROW

Subject: Programming in C

Title: Cricket Scoreboard Managing System

PROJECT REPORT

Name of Group members:
Saksham (590028288)
Ananya (590028296)
Course: B.Tech. CSE
Batch: 67
Semester: 1st
Year: 1st
Date of submission: 30/11/2025
Faculty: Mr. Vinod Kumar

ABSTRACT

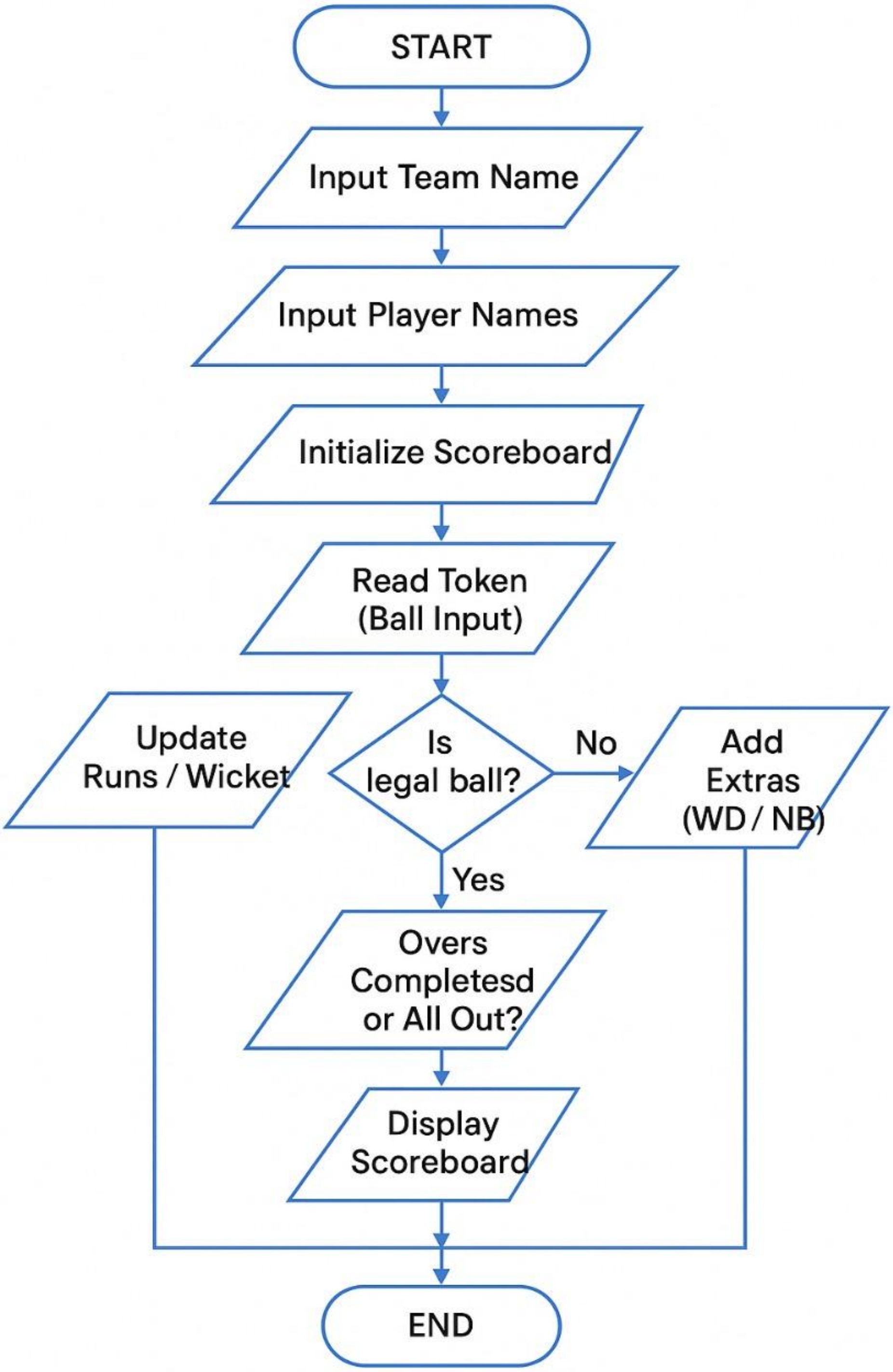
This project implements a simple, menu-driven Cricket Scoreboard Management System using the C programming language. The program allows the user to update runs, wickets, extras, and overs ball-by-ball, displaying the score dynamically. It provides essential cricket scoring features such as adding runs, wides, no-balls, byes, leg-byes, recording wickets, and resetting the scoreboard. The project demonstrates procedural programming concepts, file handling (optional), structures, and menu-driven logic.

Problem Definition

The Cricket Scoreboard Management System aims to simulate real-time cricket scoring using a C program. The objective is to create a console-based scoring tool that allows users to update runs, wickets, extras, overs, player names, and fall-of-wickets dynamically. The system also generates a comprehensive final scoreboard summarizing each player's performance. This project demonstrates structured programming, arrays, control flow, and data management through user inputs.

System Design

#Flowchart:



#Algorithm:

1. Start the program.

2. Ask the user for team name.

3. Ask for number of players.

4. Take names of all players.

5. Ask for maximum overs.

6. Initialize runs, wickets, extras, legal ball counter.

7. Set striker as Player 1 and non-striker as Player 2.

8. While legal balls < max and wickets < players - 1:

   - Display over and ball number.
   - Take input token (0–6, W, WD, NB, BY, LB).
   - If token is: -
     a. Run (0–6) *: update runs & striker stats.
     b. Wicket*: mark batsman OUT and bring next batsman.
     c. Wide/No-ball*: update extras (no ball increment for wides).
     d. Byes/Leg Byes*: update extras and legal balls.

   - Rotate strike on odd runs or end of over.

9. After innings ends, compute run rate.

10. Display the final scoreboard table.

11. End.

Problems Faced by the Group

1. **Handling Strike Rotation Logic**:

Implementing correct strike rotation after odd-numbered runs and at the end of each over required careful attention.

2. **User Input Validation**:

Ensuring the program handles tokens like WD, NB, BY, and LB without breaking the flow was challenging.

3. **Managing Player Data**:

Storing individual statistics (runs, balls, fours, sixes, out/not out) using structures needed proper indexing and updates.

4. **Fall of Wickets Tracking**:

Designing a mechanism to store fall-of-wicket details (score + over. Ball) was initially confusing.

5. **Maintaining Real-Time Scoreboard Updating**:

Making the display clean and readable while updating after every input required formatting adjustments.

6. **Handling Extras Correctly**:

Wide and no-ball rules differ from legal balls, so implementing them separately added complexity.

Cricket scoring traditionally requires manual entry of ball-by-ball updates. This project aims to simulate the scoring process through a C program that: -

* Accepts the batting team's name.

* Allows adding runs (0–6).

* Records extras such as wides and no-balls.

* Updates wickets.

* Tracks overs and balls.

* Displays the current score at any time.

* Allows resetting the scoreboard.

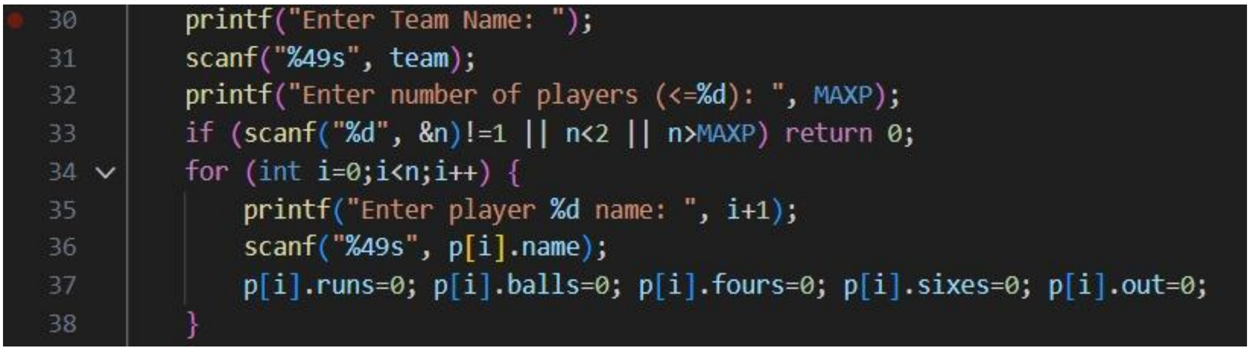
* Ends the innings automatically when 10 wickets fall.

- The objective is to create a functional, user-friendly console-based cricket scoreboard.

Implementation Details

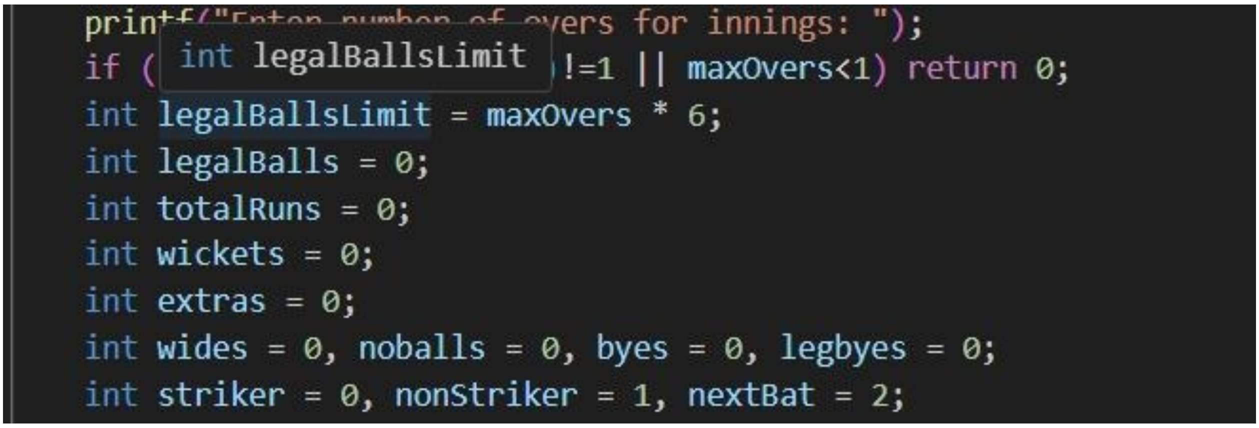
# Below are key code snippets of the system: -

1) Team Name and Player input:



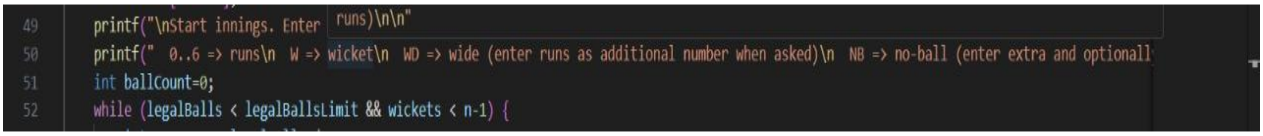
```c
    printf("Enter Team Name: ");
    scanf("%49s", team);
    printf("Enter number of players (<=%d): ", MAXP);
    if (scanf("%d", &n)!=1 || n<2 || n>MAXP) return 0;
    for (int i=0;i<n;i++) {
        printf("Enter player %d name: ", i+1);
        scanf("%49s", p[i].name);
        p[i].runs=0; p[i].balls=0; p[i].fours=0; p[i].sixes=0; p[i].out=0;
    }
```

2) Overs and Score Initialization:



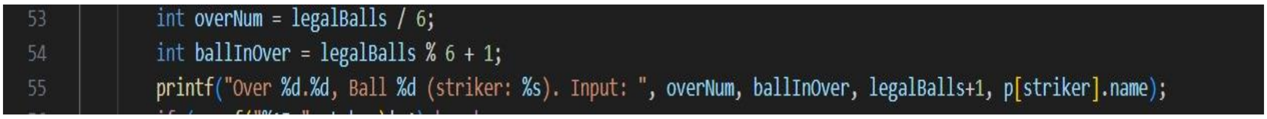
```c
    printf("Enter number of overs for innings: ");
    if ( int legalBallsLimit !=1 || maxOvers<1) return 0;
    int legalBallsLimit = maxOvers * 6;
    int legalBalls = 0;
    int totalRuns = 0;
    int wickets = 0;
    int extras = 0;
    int wides = 0, noballs = 0, byes = 0, legbyes = 0;
    int striker = 0, nonStriker = 1, nextBat = 2;
```

3) Tokens Instruction display:



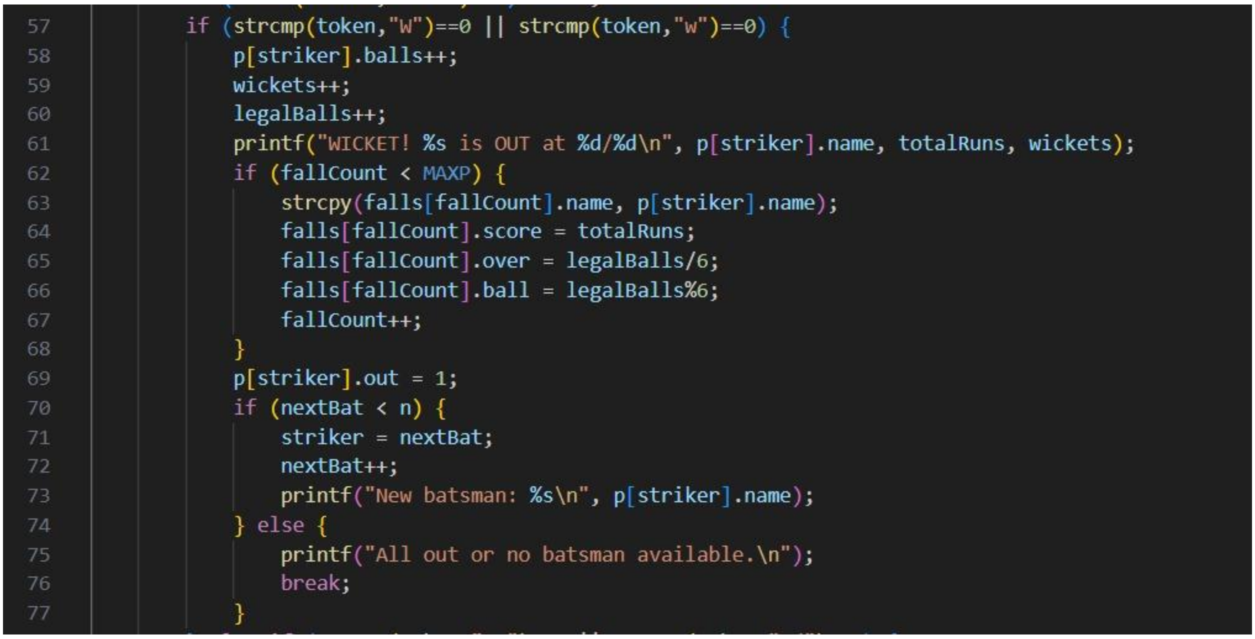
```c
    printf("\nStart innings. Enter  runs)\n\n"
    printf("  0..6 => runs\n  W => wicket\n  WD => wide (enter runs as additional number when asked)\n  NB => no-ball (enter extra and optionall
    int ballCount=0;
    while (legalBalls < legalBallsLimit && wickets < n-1) {
```

4) Display Over Ball and Strike:



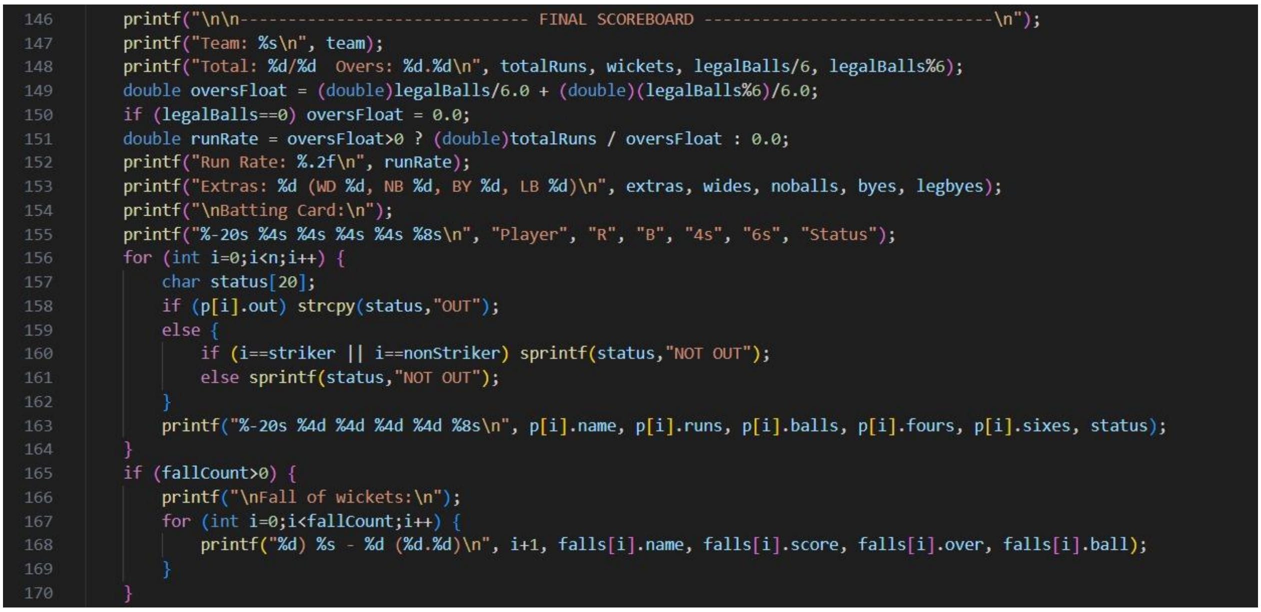
```c
int overNum = legalBalls / 6;
int ballInOver = legalBalls % 6 + 1;
printf("Over %d.%d, Ball %d (striker: %s). Input: ", overNum, ballInOver, legalBalls+1, p[striker].name);
```

5) Wicket Handling Logic:



```c
if (strcmp(token,"W")==0 || strcmp(token,"w")==0) {
    p[striker].balls++;
    wickets++;
    legalBalls++;
    printf("WICKET! %s is OUT at %d/%d\n", p[striker].name, totalRuns, wickets);
    if (fallCount < MAXP) {
        strcpy(falls[fallCount].name, p[striker].name);
        falls[fallCount].score = totalRuns;
        falls[fallCount].over = legalBalls/6;
        falls[fallCount].ball = legalBalls%6;
        fallCount++;
    }
    p[striker].out = 1;
    if (nextBat < n) {
        striker = nextBat;
        nextBat++;
        printf("New batsman: %s\n", p[striker].name);
    } else {
        printf("All out or no batsman available.\n");
        break;
    }
```

6) Final Score-Board Display:



```c
printf("\n\n------------------------------ FINAL SCOREBOARD ------------------------------\n");
printf("Team: %s\n", team);
printf("Total: %d/%d  Overs: %d.%d\n", totalRuns, wickets, legalBalls/6, legalBalls%6);
double oversFloat = (double)legalBalls/6.0 + (double)(legalBalls%6)/6.0;
if (legalBalls==0) oversFloat = 0.0;
double runRate = oversFloat>0 ? (double)totalRuns / oversFloat : 0.0;
printf("Run Rate: %.2f\n", runRate);
printf("Extras: %d (WD %d, NB %d, BY %d, LB %d)\n", extras, wides, noballs, byes, legbyes);
printf("\nBatting Card:\n");
printf("%-20s %4s %4s %4s %4s %8s\n", "Player", "R", "B", "4s", "6s", "Status");
for (int i=0;i<n;i++) {
    char status[20];
    if (p[i].out) strcpy(status,"OUT");
    else {
        if (i==striker || i==nonStriker) sprintf(status,"NOT OUT");
        else sprintf(status,"NOT OUT");
    }
    printf("%-20s %4d %4d %4d %4d %8s\n", p[i].name, p[i].runs, p[i].balls, p[i].fours, p[i].sixes, status);
}
if (fallCount>0) {
    printf("\nFall of wickets:\n");
    for (int i=0;i<fallCount;i++) {
        printf("%d) %s - %d (%d.%d)\n", i+1, falls[i].name, falls[i].score, falls[i].over, falls[i].ball);
    }
}
```

Testing and Results

#The program was tested with the following inputs:

* Team name: INDIA

* Runs added: 4, 6, 0

* Extra: wide (1)

* Wicket: 1

*Output:

1) Team: INDIA
2) Score: 11/1
3) Overs: 0.3
4) Extras: 1

- The scoreboard updated correctly after each ball, confirming that the program works as expected.

Conclusion & Future Work

#Conclusion:

This project successfully implements a functional cricket scoreboard using C. It demonstrates the use of loops, conditional statements, user input handling, and menu-driven programming. The system efficiently updates the score as per cricket rules.

#Future Enhancements:

* Add batsman strike rotation and tracking individual scores.

* Maintain bowler statistics.

* Save scoreboard to a file.

* Add a graphical user interface.

* Support second innings and match result calculation.

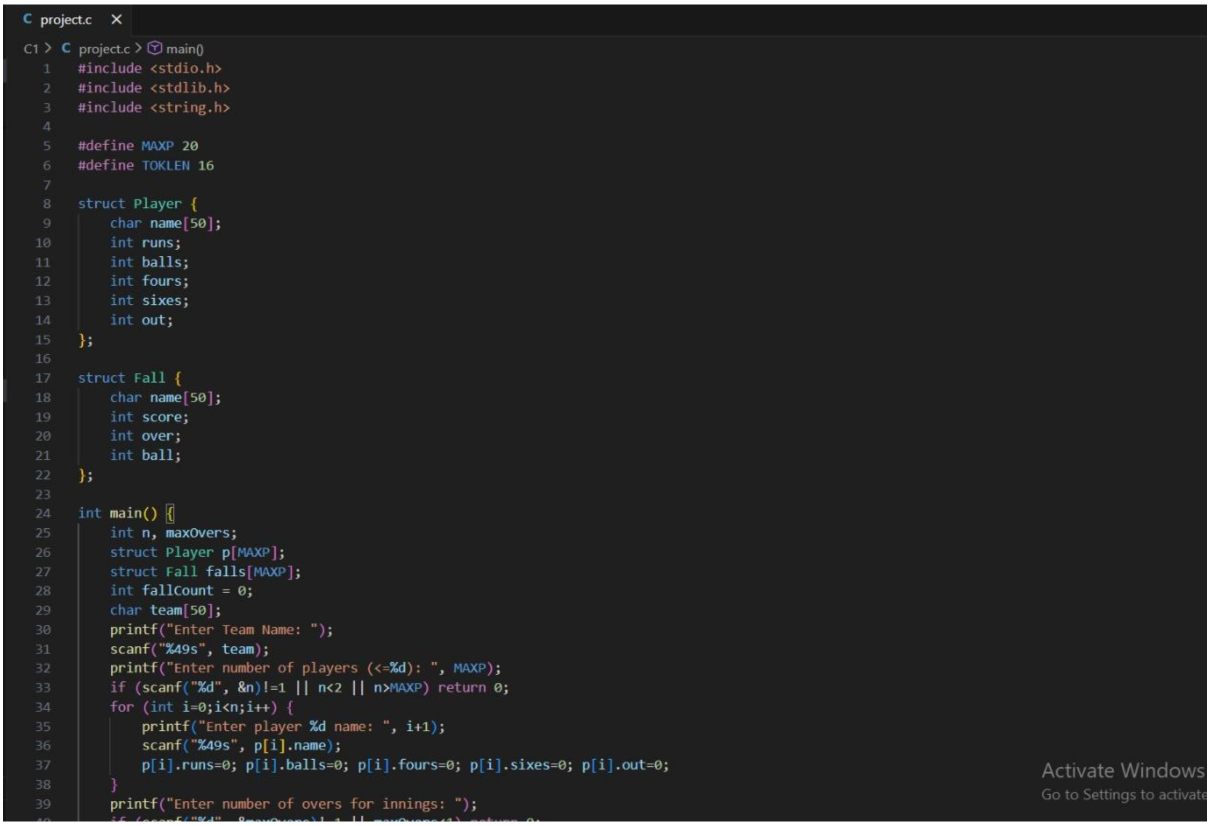
References

* Kernighan & Ritchie, The C Programming Language

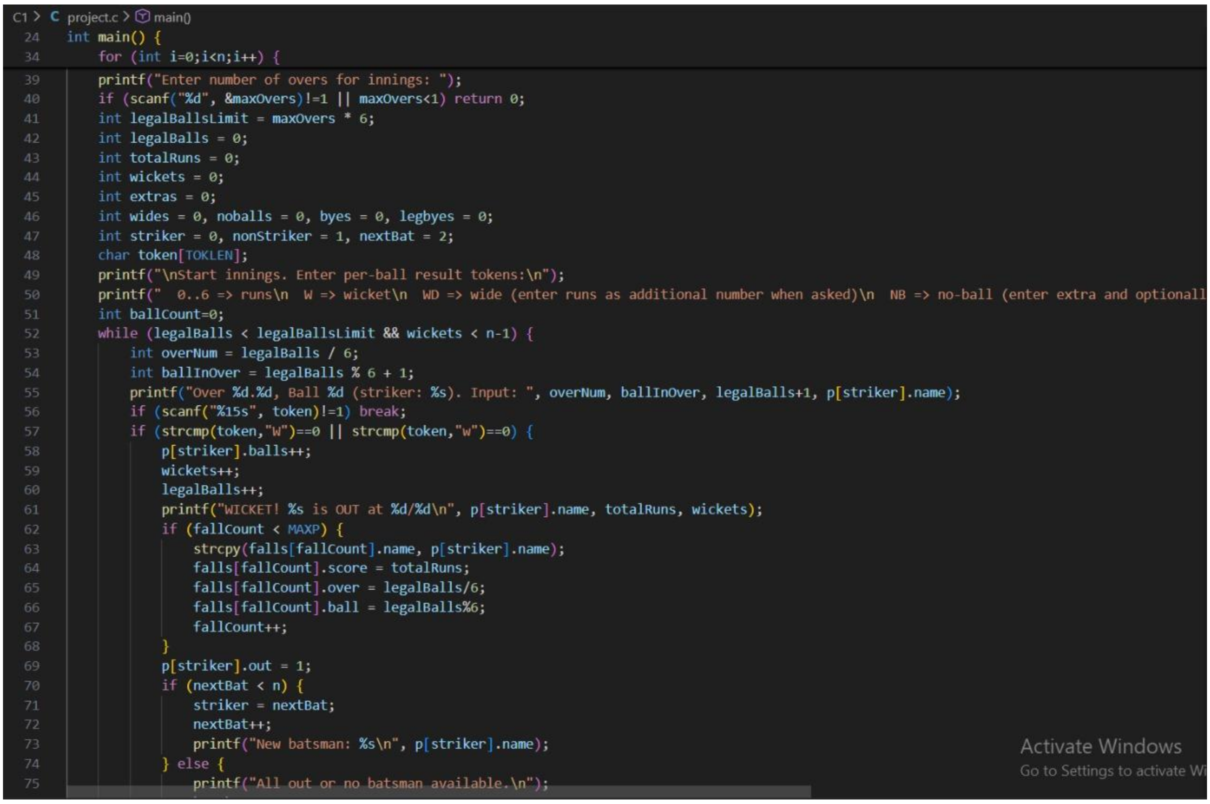
* Online C documentation (cplusplus.com, tutorialspoint.com)

* Cricket scoring guidelines (ICC resources)

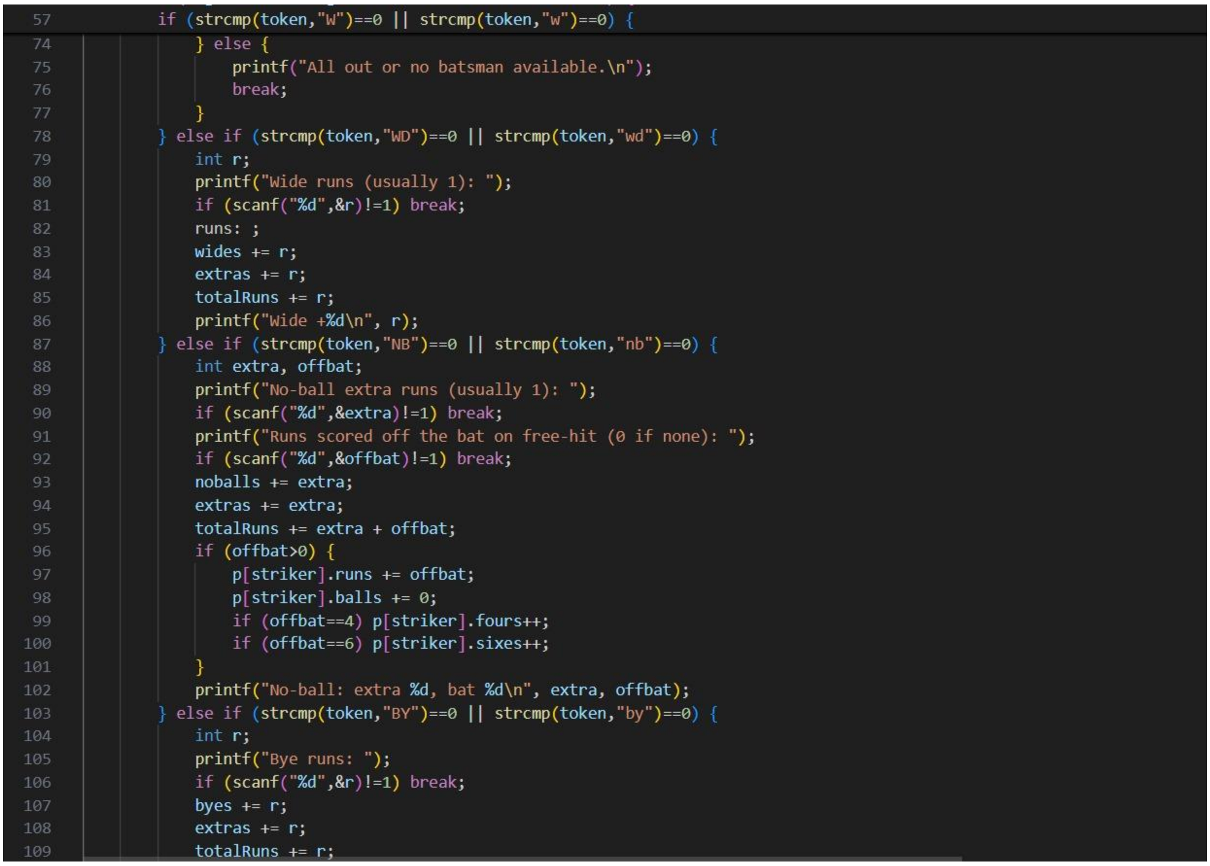
Full Source Code



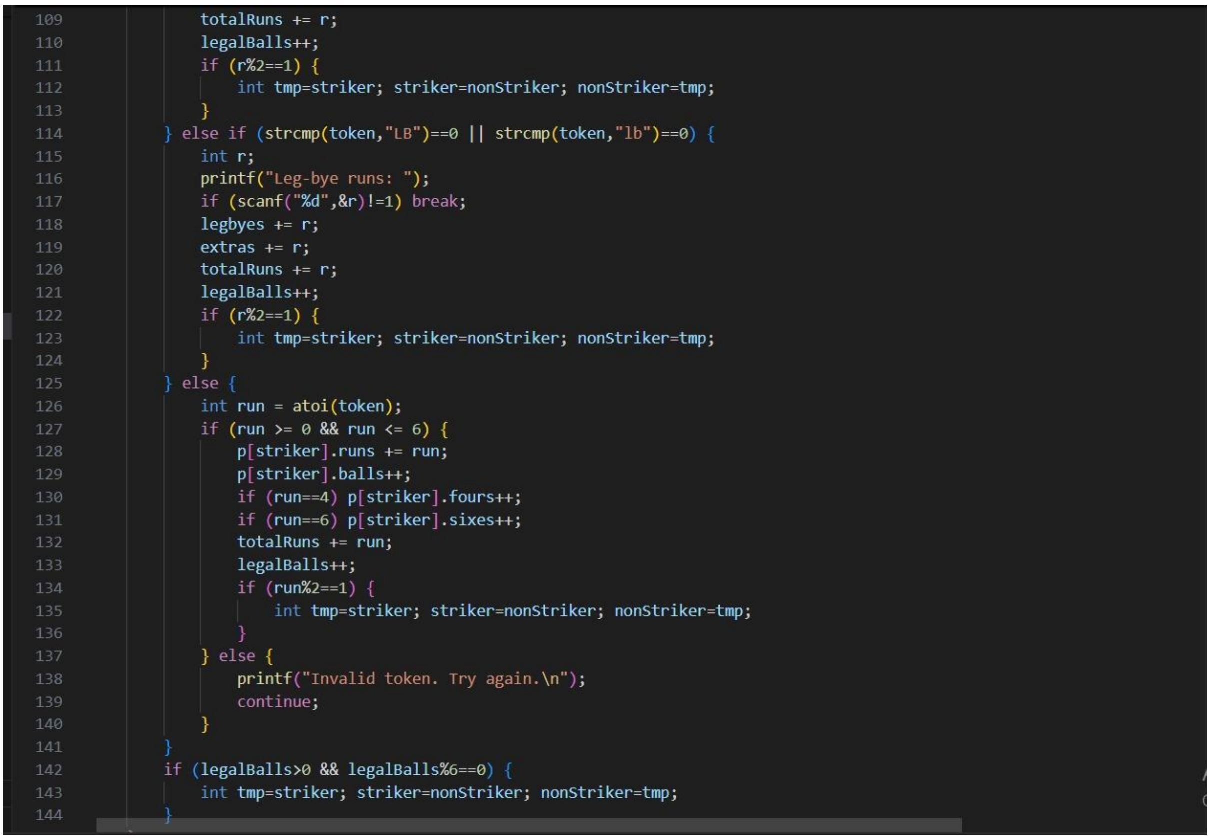
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXP 20
#define TOKLEN 16

struct Player {
    char name[50];
    int runs;
    int balls;
    int fours;
    int sixes;
    int out;
};

struct Fall {
    char name[50];
    int score;
    int over;
    int ball;
};

int main() {
    int n, maxOvers;
    struct Player p[MAXP];
    struct Fall falls[MAXP];
    int fallCount = 0;
    char team[50];
    printf("Enter Team Name: ");
    scanf("%49s", team);
    printf("Enter number of players (<=%d): ", MAXP);
    if (scanf("%d", &n)!=1 || n<2 || n>MAXP) return 0;
    for (int i=0;i<n;i++) {
        printf("Enter player %d name: ", i+1);
        scanf("%49s", p[i].name);
        p[i].runs=0; p[i].balls=0; p[i].fours=0; p[i].sixes=0; p[i].out=0;
    }
    printf("Enter number of overs for innings: ");
    if (scanf("%d", &maxOvers)!=1 || maxOvers<1) return 0;
```



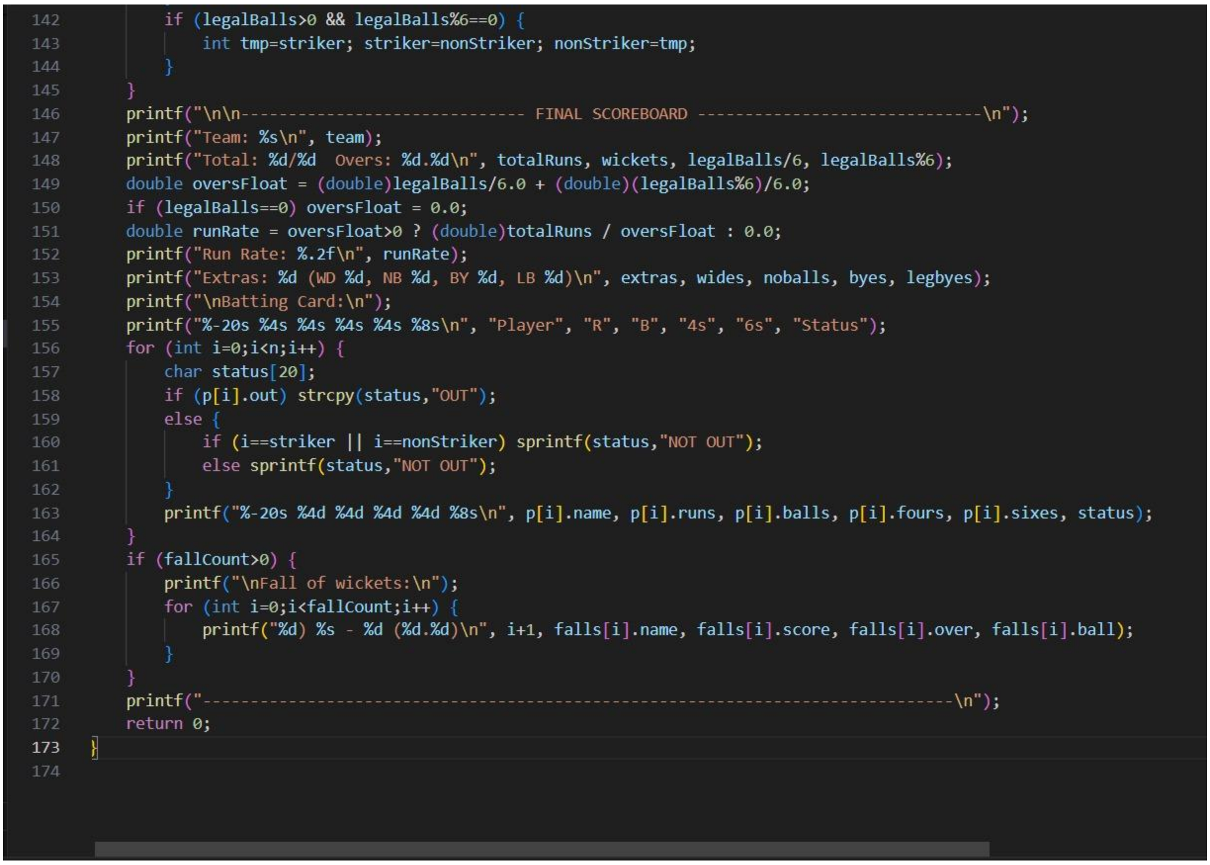
```c
int main() {
    for (int i=0;i<n;i++) {
    printf("Enter number of overs for innings: ");
    if (scanf("%d", &maxOvers)!=1 || maxOvers<1) return 0;
    int legalBallsLimit = maxOvers * 6;
    int legalBalls = 0;
    int totalRuns = 0;
    int wickets = 0;
    int extras = 0;
    int wides = 0, noballs = 0, byes = 0, legbyes = 0;
    int striker = 0, nonStriker = 1, nextBat = 2;
    char token[TOKLEN];
    printf("\nStart innings. Enter per-ball result tokens:\n");
    printf("  0..6 => runs\n  W => wicket\n  WD => wide (enter runs as additional number when asked)\n  NB => no-ball (enter extra and optionall
    int ballCount=0;
    while (legalBalls < legalBallsLimit && wickets < n-1) {
        int overNum = legalBalls / 6;
        int ballInOver = legalBalls % 6 + 1;
        printf("Over %d.%d, Ball %d (striker: %s). Input: ", overNum, ballInOver, legalBalls+1, p[striker].name);
        if (scanf("%15s", token)!=1) break;
        if (strcmp(token,"W")==0 || strcmp(token,"w")==0) {
            p[striker].balls++;
            wickets++;
            legalBalls++;
            printf("WICKET! %s is OUT at %d/%d\n", p[striker].name, totalRuns, wickets);
            if (fallCount < MAXP) {
                strcpy(falls[fallCount].name, p[striker].name);
                falls[fallCount].score = totalRuns;
                falls[fallCount].over = legalBalls/6;
                falls[fallCount].ball = legalBalls%6;
                fallCount++;
            }
            p[striker].out = 1;
            if (nextBat < n) {
                striker = nextBat;
                nextBat++;
                printf("New batsman: %s\n", p[striker].name);
            } else {
                printf("All out or no batsman available.\n");
```



```c
        if (strcmp(token,"W")==0 || strcmp(token,"w")==0) {
            } else {
                printf("All out or no batsman available.\n");
                break;
            }
        } else if (strcmp(token,"WD")==0 || strcmp(token,"wd")==0) {
            int r;
            printf("Wide runs (usually 1): ");
            if (scanf("%d",&r)!=1) break;
            runs: ;
            wides += r;
            extras += r;
            totalRuns += r;
            printf("Wide +%d\n", r);
        } else if (strcmp(token,"NB")==0 || strcmp(token,"nb")==0) {
            int extra, offbat;
            printf("No-ball extra runs (usually 1): ");
            if (scanf("%d",&extra)!=1) break;
            printf("Runs scored off the bat on free-hit (0 if none): ");
            if (scanf("%d",&offbat)!=1) break;
            noballs += extra;
            extras += extra;
            totalRuns += extra + offbat;
            if (offbat>0) {
                p[striker].runs += offbat;
                p[striker].balls += 0;
                if (offbat==4) p[striker].fours++;
                if (offbat==6) p[striker].sixes++;
            }
            printf("No-ball: extra %d, bat %d\n", extra, offbat);
        } else if (strcmp(token,"BY")==0 || strcmp(token,"by")==0) {
            int r;
            printf("Bye runs: ");
            if (scanf("%d",&r)!=1) break;
            byes += r;
            extras += r;
            totalRuns += r;
```

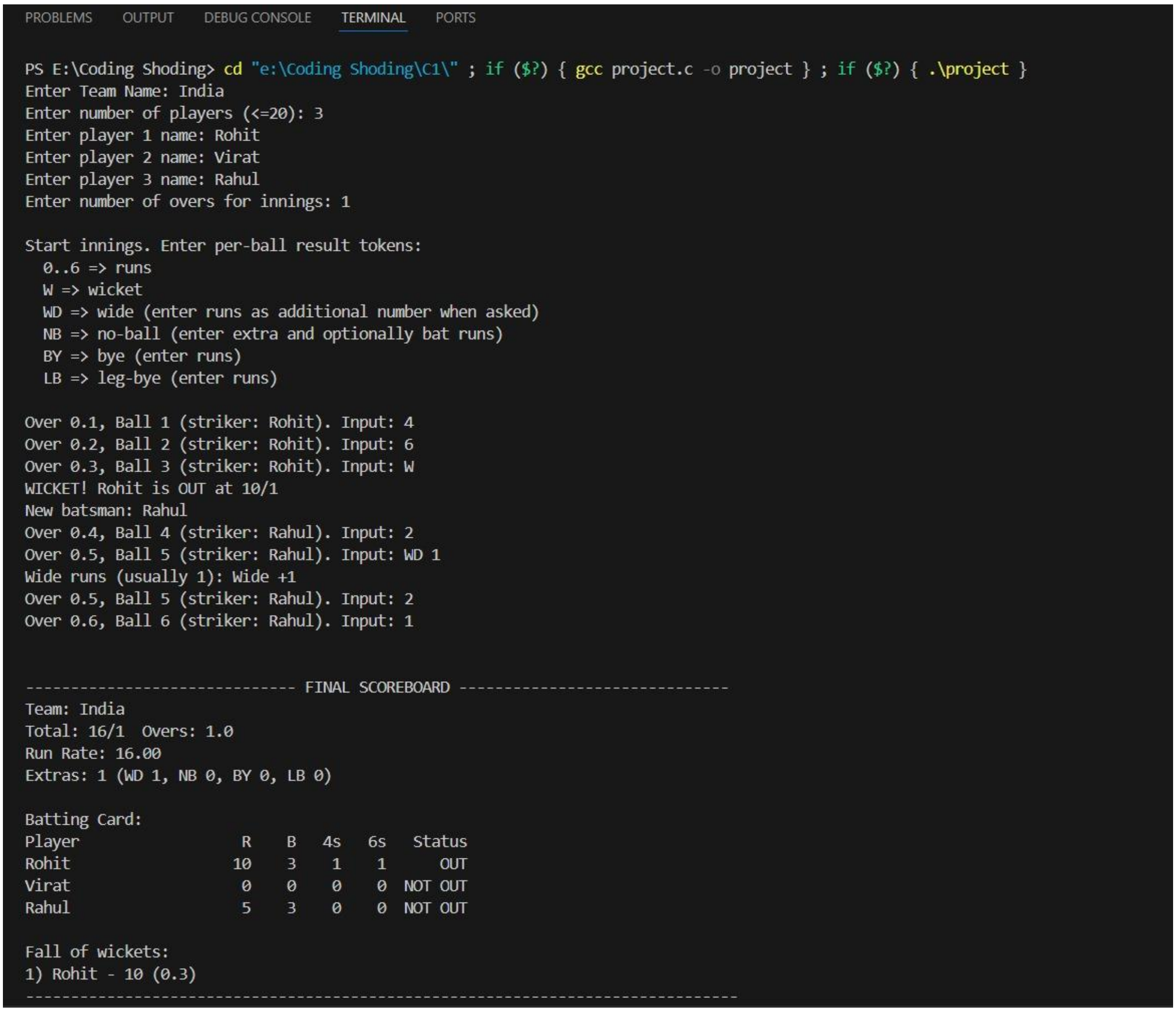


```c
            totalRuns += r;
            legalBalls++;
            if (r%2==1) {
                int tmp=striker; striker=nonStriker; nonStriker=tmp;
            }
        } else if (strcmp(token,"LB")==0 || strcmp(token,"lb")==0) {
            int r;
            printf("Leg-bye runs: ");
            if (scanf("%d",&r)!=1) break;
            legbyes += r;
            extras += r;
            totalRuns += r;
            legalBalls++;
            if (r%2==1) {
                int tmp=striker; striker=nonStriker; nonStriker=tmp;
            }
        } else {
            int run = atoi(token);
            if (run >= 0 && run <= 6) {
                p[striker].runs += run;
                p[striker].balls++;
                if (run==4) p[striker].fours++;
                if (run==6) p[striker].sixes++;
                totalRuns += run;
                legalBalls++;
                if (run%2==1) {
                    int tmp=striker; striker=nonStriker; nonStriker=tmp;
                }
            } else {
                printf("Invalid token. Try again.\n");
                continue;
            }
        }
        if (legalBalls>0 && legalBalls%6==0) {
            int tmp=striker; striker=nonStriker; nonStriker=tmp;
        }
```



```c
        if (legalBalls>0 && legalBalls%6==0) {
            int tmp=striker; striker=nonStriker; nonStriker=tmp;
        }
    }
    printf("\n\n------------------------------ FINAL SCOREBOARD ------------------------------\n");
    printf("Team: %s\n", team);
    printf("Total: %d/%d  Overs: %d.%d\n", totalRuns, wickets, legalBalls/6, legalBalls%6);
    double oversFloat = (double)legalBalls/6.0 + (double)(legalBalls%6)/6.0;
    if (legalBalls==0) oversFloat = 0.0;
    double runRate = oversFloat>0 ? (double)totalRuns / oversFloat : 0.0;
    printf("Run Rate: %.2f\n", runRate);
    printf("Extras: %d (WD %d, NB %d, BY %d, LB %d)\n", extras, wides, noballs, byes, legbyes);
    printf("\nBatting Card:\n");
    printf("%-20s %4s %4s %4s %4s %8s\n", "Player", "R", "B", "4s", "6s", "Status");
    for (int i=0;i<n;i++) {
        char status[20];
        if (p[i].out) strcpy(status,"OUT");
        else {
            if (i==striker || i==nonStriker) sprintf(status,"NOT OUT");
            else sprintf(status,"NOT OUT");
        }
        printf("%-20s %4d %4d %4d %4d %8s\n", p[i].name, p[i].runs, p[i].balls, p[i].fours, p[i].sixes, status);
    }
    if (fallCount>0) {
        printf("\nFall of wickets:\n");
        for (int i=0;i<fallCount;i++) {
            printf("%d) %s - %d (%d.%d)\n", i+1, falls[i].name, falls[i].score, falls[i].over, falls[i].ball);
        }
    }
    printf("-------------------------------------------------------------------------------\n");
    return 0;
}
```

OUTPUT



```
PS E:\Coding Shoding> cd "e:\Coding Shoding\C1\" ; if ($?) { gcc project.c -o project } ; if ($?) { .\project }
Enter Team Name: India
Enter number of players (<=20): 3
Enter player 1 name: Rohit
Enter player 2 name: Virat
Enter player 3 name: Rahul
Enter number of overs for innings: 1

Start innings. Enter per-ball result tokens:
  0..6 => runs
  W => wicket
  WD => wide (enter runs as additional number when asked)
  NB => no-ball (enter extra and optionally bat runs)
  BY => bye (enter runs)
  LB => leg-bye (enter runs)

Over 0.1, Ball 1 (striker: Rohit). Input: 4
Over 0.2, Ball 2 (striker: Rohit). Input: 6
Over 0.3, Ball 3 (striker: Rohit). Input: W
WICKET! Rohit is OUT at 10/1
New batsman: Rahul
Over 0.4, Ball 4 (striker: Rahul). Input: 2
Over 0.5, Ball 5 (striker: Rahul). Input: WD 1
Wide runs (usually 1): Wide +1
Over 0.5, Ball 5 (striker: Rahul). Input: 2
Over 0.6, Ball 6 (striker: Rahul). Input: 1


------------------------------ FINAL SCOREBOARD ------------------------------
Team: India
Total: 16/1  Overs: 1.0
Run Rate: 16.00
Extras: 1 (WD 1, NB 0, BY 0, LB 0)

Batting Card:
Player                  R    B   4s   6s   Status
Rohit                  10    3    1    1      OUT
Virat                   0    0    0    0  NOT OUT
Rahul                   5    3    0    0  NOT OUT

Fall of wickets:
1) Rohit - 10 (0.3)
-------------------------------------------------------------------------------
```

END OF REPORT