CRICKET SCORE BOARD SYSTEM

COURSE- PROGRAMMING IN C

SUBMITTED BY- Shresth Kumar

SAP ID- 590026310

INSTRUCTOR- Dr. Tanu Singh

UNIVERSITY- UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

DATE- 4/12/2025

ABSTRACT:

This project implements a fully functional cricket scoreboard system in C that tracks and displays real-time match statistics. The application is designed as a console-based interactive tool that records individual batter performance (runs, balls, fours, sixes), bowling analysis (overs, runs conceded, maidens, wickets), and comprehensive match data including extras (wide, no-balls, byes, leg-byes), fall of wickets, and partnership statistics. The system enforces cricket scoring rules and regulations, implements modular architecture with structured data types, and provides a user-friendly menu-driven interface for match administration. The application demonstrates proficiency in dynamic memory concepts, structured programming, file I/O readiness, and algorithm design through implementation of cricket scoring logic.

PROBLEM DEFINITON:

Cricket match management traditionally involves manual record-keeping on paper scorecards, which is error-prone, difficult to update in real-time, and challenging to retrieve historical statistics. There is a need for a systematic digital solution that:

- Accurately tracks all cricketers' performance metrics (runs, balls faced, boundaries)

- Records complete bowling statistics (overs bowled, maidens, runs conceded, wickets)

- Manages complex cricket-specific rules (overs structure, extras, fall of wickets)

- Displays comprehensive match scoreboard dynamically

- Enforces cricket regulations during data entry

- Handles both limited-overs and unlimited-overs formats

OBJECTIVES:

1. **To design and implement** a modular C program that manages cricket match data
2. **To enforce cricket scoring rules** including legal/illegal balls, extras, and wickets
3. **To provide real-time scoreboard** display with all relevant match statistics
4. **To implement user-friendly interface** for match operators to input match events
5. **To maintain partnership statistics** and fall of wickets for match analysis
6. **To create reusable data structures** that can be extended for other sports or reporting formats

ALGORITHM:

1. **Legal Ball Delivery:**
    a. Ball counts towards bowler's balls_bowled and batter's balls
    b. Completes overs after every 6 legal balls
    c. Strike may swap on odd runs
2. **Wide Ball:**
    a. Adds minimum 1 run to extras
    b. Does NOT count as a legal ball
    c. Does NOT count towards batter's balls faced
    d. Additional runs off wide still credited to batter
3. **No-Ball:**
    a. Adds minimum 1 run penalty
    b. Does NOT count as legal ball
    c. Additional bat runs DO count to batter and total
    d. Strike swap occurs for odd bat runs
4. **Byes/Leg-Byes:**

a. Count as legal balls (increment both bowler and batter counters)
b. Do NOT credit runs to batter, only to team extras
c. Strike swaps for odd runs
d. Count towards over completion

5. **Boundaries:**
    a. 4s and 6s tracked separately in batter statistics
    b. Always worth stated runs (4 or 6)
    c. Batter balls incremented by 1 (not by boundary value)
6. **Wicket Fall:**
    a. Records dismissal mode
    b. Updates fall of wickets array
    c. Resets partnership runs
    d. Brings next batter in at striker position
    e. Over ends if bowler completes 6 balls

OVER STRUCTURE MANAGEMENT

Legal balls: 0 1 2 3 4 5
Over completion: After 6th legal ball
Strike rotation: Automatic at end of over
Non-legal balls (wide/no-balls): Don't increment counter
Batter rotation: Manual swap for odd runs, automatic at over-end.

PARTNERSHIP CALCULATION

Current Partnership: Runs scored between current batter and non-striker since last wicket

- **Last Partnership:** Previous stand's run total (displayed when new batter arrives)
- **Reset:** On each wicket fall, current partnership resets to 0

TESTING RESULTS

1. **BASIC BATTING WITH BOUNDARIES**

Scenario: T20 match, India vs Australia, 2 batters, 1 bowler

Input Sequence:

1. Batter 1 (Virat): 4 (four)
2. Batter 2 (Rohit): 0 (dot ball)
3. Batter 1: 6 (six)
4. Batter 2: 2 (two)
5. Batter 1: 1 (one)
6. Batter 2: 0 (dot) at end of first over

Expected Output:

- Total Runs: 13 (4+0+6+2+1+0)
- Virat: 11 runs (2 balls, 1-4s, 1-6s)
- Rohit: 2 runs (3 balls)
- Strike after over: Rohit as striker (rotated)
- Overs: 1.0

2.**WICKET FALL HANDLING**

Scenario: Continuation of above match

Input Sequence:

1. Batter 2 (Rohit as striker): Wicket at' "c Fielder b Bowler"

Expected Output:

- Wickets: 1
- Rohit marked as out
- New batter (Batter 3) comes to striker position
- Fall of wickets: [13] (score when wicket fell)
- Partnership reset : partnership runs = 0

3.WIDE BALL MANAGEMENT

Scenario: Bowler bowls a wide

Input Sequence:

1. Wide: 1 run (default wide)

Expected Output:

- Extras, wide: 1

- Total Runs: +1 added
- Bowler runs conceded: +1
- Bowler balls bowled: NOT incremented (still 0.0 overs)
- Batter balls: NOT incremented (still faced 0 legal balls in this over)
- Over continues (6th legal ball not completed)

4.<u>NO BALL WITH BAT RUNS</u>

Scenario: Bowler delivers a no-ball, batter hits 2 runs

Input Sequence:

1.No-ball: 2 runs off bat

Expected Output:

- Extras Noball : 1
- Batter runs: +2 (bat runs)
- Batter balls: NOT incremented (no-ball not a legal delivery)
- Total runs: +3 (1 penalty + 2 bat runs)
- Bowler balls bowled: NOT incremented
- Strike swaps (2 is even, but actually 2 off no-ball: implementation uses batruns % 2)
- Over continues.

5.<u>BYES AND LEG-BYES</u>

Scenario: Series of defensive plays with extras

Input Sequence:

1. Byes: 3 runs
2. Leg-byes: 1 run
3. Legal 0 (dot)
4. Byes: 2 runs
5. Legal 0
6. Byes: 0 â†’ End of over

Expected Output:

- Extras. Byes: 5 (3+2+0)
- extras.legbyes: 1
- batter. Balls: 6 (all legal deliveries)
- bowler.balls_bowled: 6
- Over complete (6 legal balls)

- Strike rotates.

CONCLUSION AND FUTURE WORK

1.ACHIEVEMENTS

1. **Successfully implemented** a fully functional cricket scoreboard system that enforces real cricket rules and scoring conventions.
2. **Modular architecture** with clear separation of concerns:
    a. Data structures for players, bowling, extras
    b. Independent functions for each scoring scenario
    c. Main loop orchestrating flow
3. **Robust rule enforcement** covering:
    a. Legal vs. illegal balls
    b. Over structure (6 balls per over)
    c. Extras categories and their impact
    d. Wicket management and partnerships
    e. Strike rotation and batter sequences
4. **User-friendly interface** with menu-driven console applications, real-time scoreboard updates, and clear display of all statistics.
5. **Practical cricket knowledge** integrated: Demonstrates deep understanding of cricket scoring rules, over's structure, extras, and match management.

2.CONCEPT DEMONSTRATED

- **Structured Programming:** Multiple structures and logical function organization
- **Modular Design:** Each function handles a specific task with clear boundaries
- **Data Type Usage:** Custom struct types for complex entities
- **Array Management:** Fixed arrays for batters, bowlers, fall-of-wickets
- **Pointer Usage:** Passing structures by reference to functions
- **Control Flow:** Complex if-else logic for scoring rules
- **Input/Output:** Console I/O, formatted output, user interaction

- **Algorithm Design:** Over completion logic, partnership calculations, maiden detection

3.<u>FUTURE ENHANCEMENT</u>

1. **Two-Innings Management:**
    a. Record second innings
    b. Compare totals and determine match winner
    c. Calculate run margin or wicket margin
2. **Advanced Statistics:**
    a. Economy rate calculation (runs per over for bowlers)
    b. Strike rate calculation (runs per 100 balls for batters)
    c. Milestone tracking (50s, centuries, hat-tricks, 5-wicket hauls)
    d. Head-to-head history between teams

5. **Graphical Interface:**
    a. GUI using GTK or Qt for better UX
    b. Real-time animated scoreboard display
    c. Interactive graphics for statistics

8. **Match Commentary Generation:**
    a. Automated text commentary for each ball
    b. Match summary generation
    c. Player performance highlights

- # <u>REFERENCE</u>

1.UNIVERSITY OF PETROLEUM AND ENERGY STUDIES PROGRAMMING GUIDE
2.C PROGRAMMING TUTORIALS AND REFERENCES FOR FILE HANDLING AND STURCT USAGE.

- # <u>APPENDIX</u>

1. **Source code files (scoreboard.c , scoreboard.h)**
2. **Sample input/output screenshots**

CODE

```
#include <stdio.h>
#include <string.h>
#define MAX_PLAYERS 15
#define MAX_BOWLERS 10
#define MAX_OVERS 50
#define MAX_WICKETS 10
typedef struct {
char name[40];
int runs;
int balls;
int fours;
int sixes;
int out; // 0 not out, 1 out
char how_out[40]; // e.g., "b Bowler", "c Fielder b Bowler", "run out"
} Batter;
typedef struct {
char name[40];
int balls_bowled; // legal balls only
int runs_conceded; // includes batsmen runs + wides + no-balls + byes/leg-byes
int wickets;
int maidens;
int over_balls_in_current; // track current over balls for maiden calc
int runs_in_current_over;
```

```
} Bowler;
typedef struct {
int wides;
int noballs;
int byes;
int legbyes;
int penalty;
} Extras;
typedef struct {
int score_at_fall[MAX_WICKETS];
int wicket_no; // how many fallen
} FallOfWickets;
typedef struct {
char batting_team[40];
char bowling_team[40];
int players_per_side; // usually 11
int max_overs; // e.g., 20/50; 0 for unlimited
Batter batters[MAX_PLAYERS];
Bowler bowlers[MAX_BOWLERS];
int num_bowlers;
int striker; // index of current striker
int non_striker; // index of current non-striker
int next_batter_idx; // next unused batter index
int current_bowler; // index in bowlers[]
int total_runs;
int wickets;
int legal_balls; // total legal balls bowled
Extras extras;
FallOfWickets fow;
int partnership_runs; // current stand runs
int last_wicket_stand;// previous partnership runs
} Innings;
static void swap(int *a, int *b){ int t=*a; *a=*b; *b=t; }
static void print_overs_balls(int legal_balls){
int overs = legal_balls / 6;
int balls = legal_balls % 6;
printf("%d.%d", overs, balls);
}
static int innings_complete(Innings *inn){
int overs_limit_done = (inn->max_overs>0) && ((inn->legal_balls/6) >= inn->max_overs);
int all_out = inn->wickets >= (inn->players_per_side - 1);
```

```
return overs_limit_done || all_out;
}
static void start_innings(Innings *inn){
memset(inn, 0, sizeof(*inn));
printf("Batting team: "); fgets(inn->batting_team, sizeof(inn->batting_team), stdin);
printf("Bowling team: "); fgets(inn->bowling_team, sizeof(inn->bowling_team), stdin);
// strip newlines
inn->batting_team[strcspn(inn->batting_team, "\n")] = 0;
inn->bowling_team[strcspn(inn->bowling_team, "\n")] = 0;

printf("Players per side (<=%d, default 11): ", MAX_PLAYERS);
int n; if (scanf("%d", &n)!=1 || n<=0 || n>MAX_PLAYERS) n=11;
inn->players_per_side = n;

printf("Max overs (0 for unlimited, <=%d): ", MAX_OVERS);
int mo; if (scanf("%d", &mo)!=1 || mo<0 || mo>MAX_OVERS) mo=0;
inn->max_overs = mo;

// read batting names
getchar(); // consume newline
for(int i=0;i<inn->players_per_side;i++){
printf("Batter %d name: ", i+1);
fgets(inn->batters[i].name, sizeof(inn->batters[i].name), stdin);
inn->batters[i].name[strcspn(inn->batters[i].name, "\n")]=0;
inn->batters[i].runs=inn->batters[i].balls=inn->batters[i].fours=inn->batters[i].sixes=0;
inn->batters[i].out=0;
strcpy(inn->batters[i].how_out, "not out");
}

printf("Number of bowlers to rotate (<=%d): ", MAX_BOWLERS);
int nb; if (scanf("%d",&nb)!=1 || nb<=0 || nb>MAX_BOWLERS) nb=5;
getchar();
inn->num_bowlers = nb;
for(int i=0;i<nb;i++){
printf("Bowler %d name: ", i+1);
fgets(inn->bowlers[i].name, sizeof(inn->bowlers[i].name), stdin);
inn->bowlers[i].name[strcspn(inn->bowlers[i].name, "\n")] = 0;
inn->bowlers[i].balls_bowled=inn->bowlers[i].runs_conceded=inn->bowlers[i].wickets=inn-
>bowlers[i].maidens=0;
inn->bowlers[i].over_balls_in_current=0;
inn->bowlers[i].runs_in_current_over=0;
```

```
}

inn->striker = 0;
inn->non_striker = 1;
inn->next_batter_idx = 2;
inn->current_bowler = 0;
inn->total_runs=0;
inn->wickets=0;
inn->legal_balls=0;
memset(&inn->extras, 0, sizeof(inn->extras));
inn->fow.wicket_no=0;
inn->partnership_runs=0;
inn->last_wicket_stand=0;

printf("\nInnings started: %s vs %s\n", inn->batting_team, inn->bowling_team);
}

static void end_of_over_rotate(Innings *inn){
// Over complete: swap strike
swap(&inn->striker, &inn->non_striker);
// check maiden
Bowler *bw = &inn->bowlers[inn->current_bowler];
if (bw->runs_in_current_over == 0) bw->maidens++;
bw->runs_in_current_over = 0;
bw->over_balls_in_current = 0;

// choose next bowler (not same consecutive)
printf("Select next bowler index (1-%d, not same as last %d: %s): ",
inn->num_bowlers, inn->current_bowler+1, inn->bowlers[inn->current_bowler].name);
int idx; if (scanf("%d",&idx)!=1) idx=1;
idx = (idx<1?1:idx>inn->num_bowlers?inn->num_bowlers:idx);
if (idx-1 == inn->current_bowler){
// force rotate to next different bowler
idx = (idx % inn->num_bowlers) + 1;
}
inn->current_bowler = idx-1;
getchar();
}

static void record_boundary(Batter *bt, int runs){
bt->runs += runs;
```

```
bt->balls += 1;
if (runs==4) bt->fours++;
if (runs==6) bt->sixes++;
}

static void show_score(Innings *inn){
printf("\n---------- SCOREBOARD ----------\n");
printf("%s: %d/%d in ", inn->batting_team, inn->total_runs, inn->wickets);
print_overs_balls(inn->legal_balls);
printf(" overs\n");
printf("Extras: Wd %d, NB %d, B %d, LB %d, Pen %d | Total extras: %d\n",
inn->extras.wides, inn->extras.noballs, inn->extras.byes,
inn->extras.legbyes, inn->extras.penalty,
inn->extras.wides + inn->extras.noballs + inn->extras.byes + inn->extras.legbyes + inn-
>extras.penalty);
printf("Partnership: %d | Last wicket stand: %d\n", inn->partnership_runs, inn-
>last_wicket_stand);

printf("\nBatting:\n");
for(int i=0;i<inn->players_per_side;i++){
Batter *b = &inn->batters[i];
if (i==inn->striker) printf("-> ");
else if (i==inn->non_striker) printf(" * ");
else printf("  ");
printf("%-20s %3d (%d) 4s:%d 6s:%d | %s\n",
b->name, b->runs, b->balls, b->fours, b->sixes, b->how_out);
}

printf("\nBowling:\n");
for(int i=0;i<inn->num_bowlers;i++){
Bowler *bw = &inn->bowlers[i];
int overs = bw->balls_bowled / 6;
int balls = bw->balls_bowled % 6;
printf("%-20s %d.%d- M:%d - R:%d - W:%d\n",
bw->name, overs, balls, bw->maidens, bw->runs_conceded, bw->wickets);
}

printf("\nFall of wickets: ");
if (inn->fow.wicket_no==0) printf("-\n");
else {
for(int i=0;i<inn->fow.wicket_no;i++){
```

```
printf("%d", inn->fow.score_at_fall[i]);
if (i<inn->fow.wicket_no-1) printf(", ");
}
printf("\n");
}
printf("-------------------------------------\n\n");
}

static void wicket_falls(Innings *inn, const char *how_out){
int out_batter = inn->striker; // assume striker by default
// Mark out
inn->batters[out_batter].out = 1;
strncpy(inn->batters[out_batter].how_out, how_out, sizeof(inn->batters[out_batter].how_out)-1);
inn->wickets++;
inn->fow.score_at_fall[inn->fow.wicket_no++] = inn->total_runs;
inn->last_wicket_stand = inn->partnership_runs;
inn->partnership_runs = 0;

// credit ball to batter and bowler for legal wicket
inn->batters[out_batter].balls += 1;
Bowler *bw = &inn->bowlers[inn->current_bowler];
bw->balls_bowled += 1;
bw->over_balls_in_current += 1;
// add wicket to bowler unless it's run out/retired
if (strncmp(how_out, "run out", 7)!=0 && strncmp(how_out, "retired", 7)!=0)
bw->wickets++;

inn->legal_balls++;

// new batter comes in at striker
if (inn->next_batter_idx < inn->players_per_side){
inn->striker = inn->next_batter_idx++;
} else {
// all out
}

// end of over check
if (bw->over_balls_in_current == 6){
end_of_over_rotate(inn);
}
}
```

```
static void ball_menu(){
printf("Ball outcome:\n");
printf(" 1) 0 runs\n 2) 1 run\n 3) 2 runs\n 4) 3 runs\n 5) Four (4)\n 6) Six (6)\n");
printf(" 7) Wicket\n 8) Wide (+1, add more if run as wides)\n 9) No-ball (+1) with bat runs\n");
printf("10) Byes (0-6)\n11) Leg-byes (0-6)\n12) Penalty runs (+5)\n13) Swap strike (end of odd
runs or between overs)\n");
}

static void process_legal_runs(Innings *inn, int runs){
Batter *bt = &inn->batters[inn->striker];
Bowler *bw = &inn->bowlers[inn->current_bowler];

// Record to batter
if (runs==4 || runs==6) record_boundary(bt, runs);
else { bt->runs += runs; bt->balls += 1; }

// Bowling stats
bw->runs_conceded += runs;
bw->balls_bowled += 1;
bw->over_balls_in_current += 1;
bw->runs_in_current_over += runs;

// Team and partnerships
inn->total_runs += runs;
inn->partnership_runs += runs;
inn->legal_balls++;

// strike swap on odd runs
if (runs % 2 == 1) swap(&inn->striker, &inn->non_striker);

// end of over
if (bw->over_balls_in_current == 6){
end_of_over_rotate(inn);
}
}

static void process_wide(Innings *inn, int wide_runs){
// wide_runs >=1 (includes the 1 penalty); no legal ball
Bowler *bw = &inn->bowlers[inn->current_bowler];
inn->extras.wides += wide_runs;
```

```
inn->total_runs += wide_runs;
bw->runs_conceded += wide_runs;
// no increment of balls_bowled or legal balls
}

static void process_noball_with_bat_runs(Innings *inn, int bat_runs){
// +1 no-ball penalty; no legal ball yet
Bowler *bw = &inn->bowlers[inn->current_bowler];
Batter *bt = &inn->batters[inn->striker];

inn->extras.noballs += 1;
inn->total_runs += 1;
bw->runs_conceded += 1;

// runs off the bat also add to total and batter, but ball not legal
if (bat_runs>0){
if (bat_runs==4 || bat_runs==6){
// boundaries off no-ball add bat runs
bt->runs += bat_runs;
if (bat_runs==4) bt->fours++;
if (bat_runs==6) bt->sixes++;
} else {
bt->runs += bat_runs;
}
inn->total_runs += bat_runs;
inn->partnership_runs += bat_runs;
bw->runs_conceded += bat_runs;
// strike swap for odd bat_runs on no-ball does happen
if (bat_runs % 2 == 1) swap(&inn->striker, &inn->non_striker);
}
}

static void process_byes(Innings *inn, int runs, int is_legbye){
// counts as extras; not a legal ball? Actually byes/leg-byes occur on legal deliveries. They DO
count as a legal ball to bowler and batter faces the ball but no bat runs.
Bowler *bw = &inn->bowlers[inn->current_bowler];
Batter *bt = &inn->batters[inn->striker];

if (is_legbye) inn->extras.legbyes += runs;
else inn->extras.byes += runs;
```

```
inn->total_runs += runs;
inn->partnership_runs += runs;
bw->runs_conceded += runs;

// credit ball faced to batter
bt->balls += 1;
bw->balls_bowled += 1;
bw->over_balls_in_current += 1;
inn->legal_balls++;

// strike swap for odd runs
if (runs % 2 == 1) swap(&inn->striker, &inn->non_striker);

if (bw->over_balls_in_current==6){
end_of_over_rotate(inn);
}
}

static void process_penalty(Innings *inn, int runs){
inn->extras.penalty += runs;
inn->total_runs += runs;
// Penalty runs are added to total; whether they add to bowler runs varies by context; keep
simple: add to team only.
}

static void main_loop(Innings *inn){
while (!innings_complete(inn)){
show_score(inn);
printf("Striker: %s | Non-striker: %s | Bowler: %s\n",
inn->batters[inn->striker].name,
inn->batters[inn->non_striker].name,
inn->bowlers[inn->current_bowler].name);
ball_menu();
printf("Choose: ");
int ch; if (scanf("%d",&ch)!=1){ getchar(); continue; }

if (ch>=1 && ch<=6){
int runs = 0;
if (ch==1) runs=0;
else if (ch==2) runs=1;
else if (ch==3) runs=2;
```

```
else if (ch==4) runs=3;
else if (ch==5) runs=4;
else if (ch==6) runs=6;
process_legal_runs(inn, runs);
} else if (ch==7){
getchar();
char how[40];
printf("How out? (e.g., b %s, c Fielder b , run out): ", inn->bowlers[inn->current_bowler].name);
fgets(how, sizeof(how), stdin);
how[strcspn(how, "\n")]=0;
wicket_falls(inn, how);
if (innings_complete(inn)) break;
} else if (ch==8){
int r; printf("Wide runs (>=1, include all run(s) as wides): "); scanf("%d",&r);
if (r<1) r=1;
process_wide(inn, r);
} else if (ch==9){
int r; printf("Runs off the bat on no-ball (0/1/2/3/4/6): "); scanf("%d",&r);
if (r<0) r=0;
process_noball_with_bat_runs(inn, r);
} else if (ch==10){
int r; printf("Byes runs (0-6): "); scanf("%d",&r);
if (r<0) r=0; if (r>6) r=6;
process_byes(inn, r, 0);
} else if (ch==11){
int r; printf("Leg-byes runs (0-6): "); scanf("%d",&r);
if (r<0) r=0; if (r>6) r=6;
process_byes(inn, r, 1);
} else if (ch==12){
process_penalty(inn, 5);
} else if (ch==13){
swap(&inn->striker, &inn->non_striker);
} else {
printf("Invalid.\n");
}
}

printf("\nInnings complete.\n");
show_score(inn);
}
```

```
int main(void){
Innings inn;
start_innings(&inn);
main_loop(&inn);
return 0;
}
```

OUTPUT

```
shresthkumar@shresths-MacBook-Air ~ %   gcc majorproject.c
shresthkumar@shresths-MacBook-Air ~ % ./majorproject
  Batting team: INDIA
  Bowling team: AUSTRALIA
  Players per side (<=15, default 11): 5
  Max overs (0 for unlimited, <=50): 5
  Batter 1 name: ROHIT SHARMA
  Batter 2 name: VIRAT KOHLI
  Batter 3 name: SURESH RAINA
  Batter 4 name: SHIKHAR DHAWAN
  Batter 5 name: MS DHONI
  Number of bowlers to rotate (<=10): 4
  Bowler 1 name: MITCHEL STARC
  Bowler 2 name: PAT CUMMIND
  Bowler 3 name: ADAM ZAMPA
  Bowler 4 name: NATHAN ELLIS
```

```
Innings started: INDIA vs AUSTRALIA

------------ SCOREBOARD ------------
INDIA: 0/0 in 0.0 overs
Extras: Wd 0, NB 0, B 0, LB 0, Pen 0  | Total extras: 0
Partnership: 0  | Last wicket stand: 0

Batting:
-> ROHIT SHARMA           0 (0)  4s:0 6s:0  | not out
 * VIRAT KOHLI            0 (0)  4s:0 6s:0  | not out
   SURESH RAINA           0 (0)  4s:0 6s:0  | not out
   SHIKHAR DHAWAN         0 (0)  4s:0 6s:0  | not out
   MS DHONI               0 (0)  4s:0 6s:0  | not out

Bowling:
MITCHEL STARC        0.0- M:0 - R:0 - W:0
PAT CUMMIND          0.0- M:0 - R:0 - W:0
ADAM ZAMPA           0.0- M:0 - R:0 - W:0
NATHAN ELLIS         0.0- M:0 - R:0 - W:0

Fall of wickets: -
--------------------------------------

Striker: ROHIT SHARMA | Non-striker: VIRAT KOHLI | Bowler: MITCHEL STARC
Ball outcome:
 1) 0 runs
 2) 1 run
 3) 2 runs
 4) 3 runs
 5) Four (4)
 6) Six (6)
 7) Wicket
 8) Wide (+1, add more if run as wides)
 9) No-ball (+1) with bat runs
10) Byes (0-6)
11) Leg-byes (0-6)
12) Penalty runs (+5)
13) Swap strike (end of odd runs or between overs)
Choose: 2
```

```
------------ SCOREBOARD ------------
INDIA: 1/0 in 0.1 overs
Extras: Wd 0, NB 0, B 0, LB 0, Pen 0  | Total extras: 0
Partnership: 1  | Last wicket stand: 0

Batting:
 * ROHIT SHARMA           1 (1)  4s:0 6s:0  | not out
-> VIRAT KOHLI            0 (0)  4s:0 6s:0  | not out
   SURESH RAINA           0 (0)  4s:0 6s:0  | not out
   SHIKHAR DHAWAN         0 (0)  4s:0 6s:0  | not out
   MS DHONI               0 (0)  4s:0 6s:0  | not out

Bowling:
MITCHEL STARC        0.1- M:0 - R:1 - W:0
PAT CUMMIND          0.0- M:0 - R:0 - W:0
ADAM ZAMPA           0.0- M:0 - R:0 - W:0
NATHAN ELLIS         0.0- M:0 - R:0 - W:0

Fall of wickets: -
------------------------------------

Striker: VIRAT KOHLI | Non-striker: ROHIT SHARMA | Bowler: MITCHEL STARC
Ball outcome:
 1) 0 runs
 2) 1 run
 3) 2 runs
 4) 3 runs
 5) Four (4)
 6) Six (6)
 7) Wicket
 8) Wide (+1, add more if run as wides)
 9) No-ball (+1) with bat runs
10) Byes (0-6)
11) Leg-byes (0-6)
12) Penalty runs (+5)
13) Swap strike (end of odd runs or between overs)
```

```
Striker: VIRAT KOHLI | Non-striker: ROHIT SHARMA | Bowler: MITCHEL STARC
Ball outcome:
 1) 0 runs
 2) 1 run
 3) 2 runs
 4) 3 runs
 5) Four (4)
 6) Six (6)
 7) Wicket
 8) Wide (+1, add more if run as wides)
 9) No-ball (+1) with bat runs
10) Byes (0-6)
11) Leg-byes (0-6)
12) Penalty runs (+5)
13) Swap strike (end of odd runs or between overs)
Choose: 7
How out? (e.g., b MITCHEL STARC, c Fielder b , run out): b starc cummins

------------ SCOREBOARD ------------
INDIA: 1/1 in 0.2 overs
Extras: Wd 0, NB 0, B 0, LB 0, Pen 0  | Total extras: 0
Partnership: 0  | Last wicket stand: 1

Batting:
 * ROHIT SHARMA           1 (1)  4s:0 6s:0  | not out
   VIRAT KOHLI            0 (1)  4s:0 6s:0  | b starc cummins
-> SURESH RAINA           0 (0)  4s:0 6s:0  | not out
   SHIKHAR DHAWAN         0 (0)  4s:0 6s:0  | not out
   MS DHONI               0 (0)  4s:0 6s:0  | not out

Bowling:
MITCHEL STARC        0.2- M:0 - R:1 - W:1
PAT CUMMIND          0.0- M:0 - R:0 - W:0
ADAM ZAMPA           0.0- M:0 - R:0 - W:0
NATHAN ELLIS         0.0- M:0 - R:0 - W:0

Fall of wickets: 1
------------------------------------
```