Capgemini

[redacted]

High Level Design & Low Level Design

Index

1. Introduction

The bowling scorer application is a system that allows the user to automatically store the scores for each and every player in a systematic order inside the database removing the traditional way of using scorekeeping where every player had to keep their own scores. Now with the introduction of this system it simplifies the process as now every player record is accessible at a single click right from their names to previous tournaments won to how much the player has scored in every single frame throughout the tournament and calculating the winner at the end of the day on the basis of total score of the players.

1.1 Intended Audience: -

The target audience set for this project can be identified as an organizer who is conducting a bowling tournament and automate the process of handling the tournament which includes maintaining records of the players and their scores,

1.2 Project Purpose: -

The bowling game is a project that helps us understand the basic concepts of functions, file handling, and data structure. The automatic bowling scorer will take bowler id from the bowler. It will create a scoresheet for the entire tournament. It will record all the tournament details and who won in the game. If the bowler is new then it will save in the bowler's record. We can maintain a database of bowlers by adding, displaying, updating and removing by the main menu of this Bowling Game program.

1.3 Key Project Objectives: -

a. Allow the Bowler to enter the Bowler ID

b. Play the game

c. updating the scores for each frame

d. Displays all the records of Bowler

e. Modify/Update the Bowle records.

f. calculating the cumulative score.

1.4 Project scope : -

This project aims to create the development of an Automatic Bowling score application,Which takes the bowler information such as ID and Name, adds it to the database and processes the score of the player after the game and adds it to the Bowling data of the user as how many tournaments won along with also displaying the bowling game score of the user to which frame the user is currently bowling in and what is the total score after each frame.

2. Design Overview: -

- **Bowling Game comprises of the following modules in maintain bowler database:**

Name of the Module	Add Module
Handled by	
Description	The bowler adds the record in the database

Name of the Module	Delete Module
Handled by	
Description	The bowler deletes a record from database

Name of the Module	View Module
Handled by	
Description	The bowler views the record from database

Name of the Module	Edit Module
Handled by	
Description	The bowler edits the record in the database

- **Bowling Game comprises of the following modules in play the game:**

Name of the Module	Random Number module
Handled by	
Description	Generating the random numbers

Name of the Module	Strike Condition Module
Handled by	
Description	Knocking down all the pins with the first ball ends the frame

Name of the Module	Spare Condition Module
Handled by	
Description	Knocking down all the pins with the first two balls

Name of the Module	Extra balls Module
Handled by	
Description	A spare in the tenth frame earns one extra ball

Name of the Module	Play the game Module
Handled by	
Description	Bowler will play the game and awarded with scores

- **Bowling Game comprises of the following modules in show reports:**

Name of the Module	Bowler datasheet Module
Handled by	
Description	It will show the bowler datasheet from bowler database

Name of the Module	Bowling day report Module
Handled by	
Description	It will display the bowling day report from bowling database

2.1 Design Objectives:

1. Add different bowler profiles to the records.
2. Start the scoring application.
3. Updating the scores for each frame and the total score.
4. Displays all the records of Bowler.
5. Modify/Update the Bowler records
6. Calculating the cumulative score.

2.2 Design Alternative: -

We have used a linked list structure to store data i.e.. Bowler ID, Name, Years of Experience and Number of Tournaments won, Frame no, balls, score.

2.3 User Interface Paradigms: -

The Bowling game provides an option to Bowlers by generating the score automatically in each frame while playing and keeping the records of players also.

2.4 Validation: -

- Bowler Id should not be blank and Duplication is not allowed and characters aren't allowed in the ID.
- In case of integer validation, if the entered Bowler Id is not Integer it displays the message ID must contain Integer only and should not contain more than 15 digits.
- We check for the validity of the name; it should not contain more than 15 characters and Bowler name should not be empty and only alphabets.

3. SYSTEM ARCHITECTURE: -

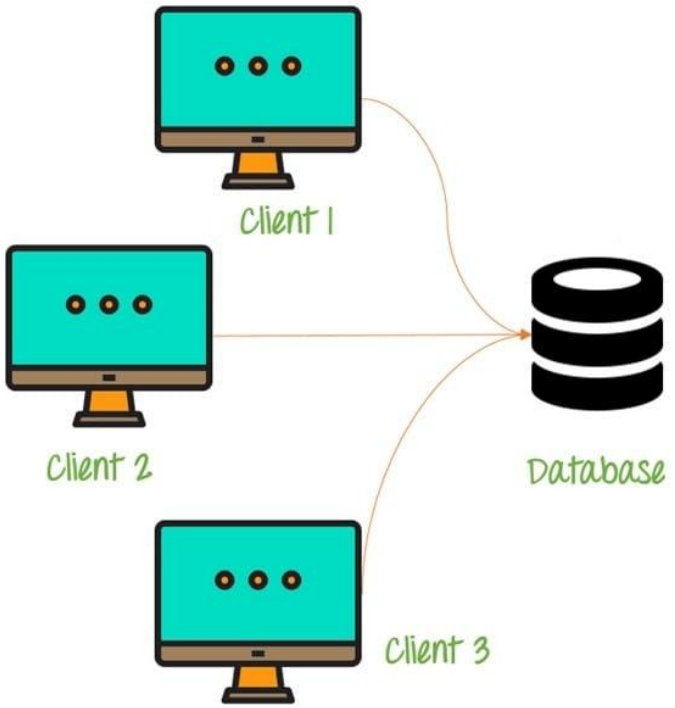
3.1. Database Architecture

The architecture used in this system comprises of the database architecture. It is a representation of the database management system design, wherein you can design, develop, implement and maintain the database. This architecture allows dividing the database into different components that can be independently modified, changed, replaced and altered as required for the system.

The database architecture is divided into three tiers namely:-

- 1 - Tier Architecture
- 2 - Tier Architecture
- 3 - Tier Architecture

Our system is based on the Tier 1 model of the database architecture. In this type of model the database is directly available to the user, the user can directly access the database and all of its contents. Which enables the user to directly interact and execute operations.



Some of the characteristics of Database Architecture are:

Self-Describing Nature of a Database System :

- One of the most fundamental characteristics of the database approach is that the database system contains not only the database itself but also an entire definition or description of the database structure and constraints also known as metadata of the database.

Isolation between Data, Programs and Data Abstraction:

- In a traditional file processing system, the structure of database knowledge files is embedded within the application programs, so any changes to the structure of a file may require changing all programs that access that file.

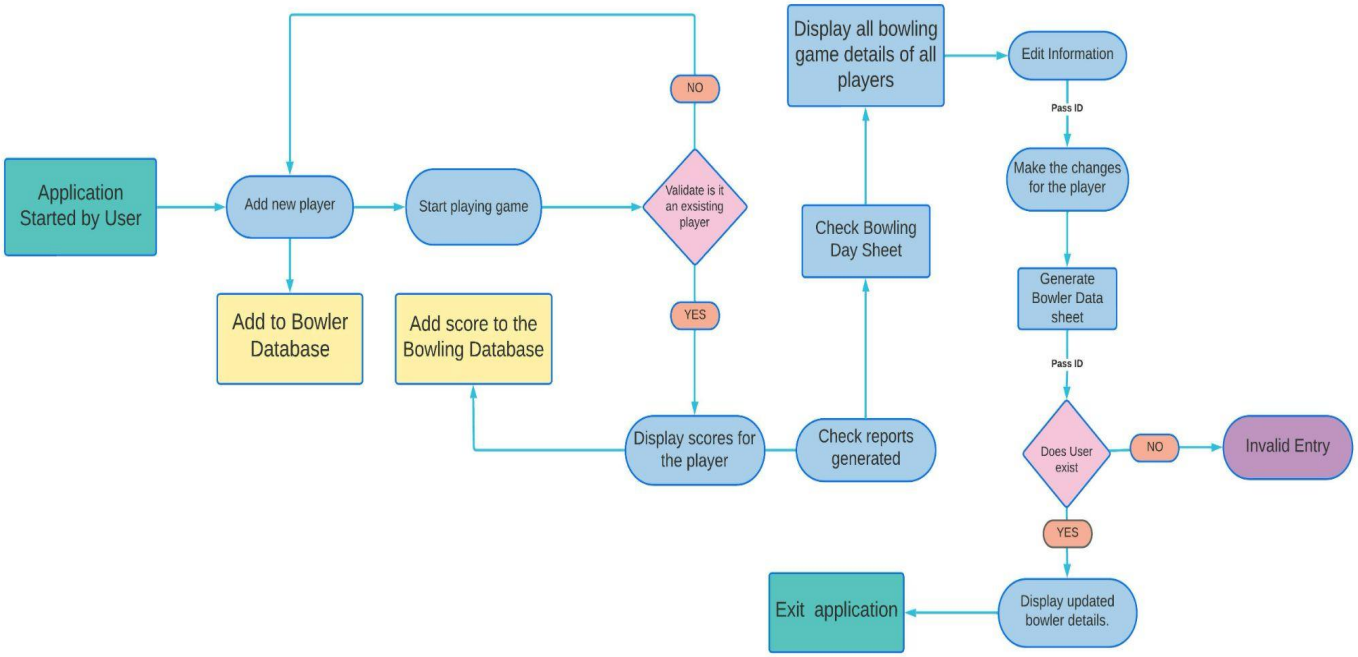
Support for Multiple Views of the Data :

- A database sometimes has many users, each of whom may require a special perspective or view of the database.

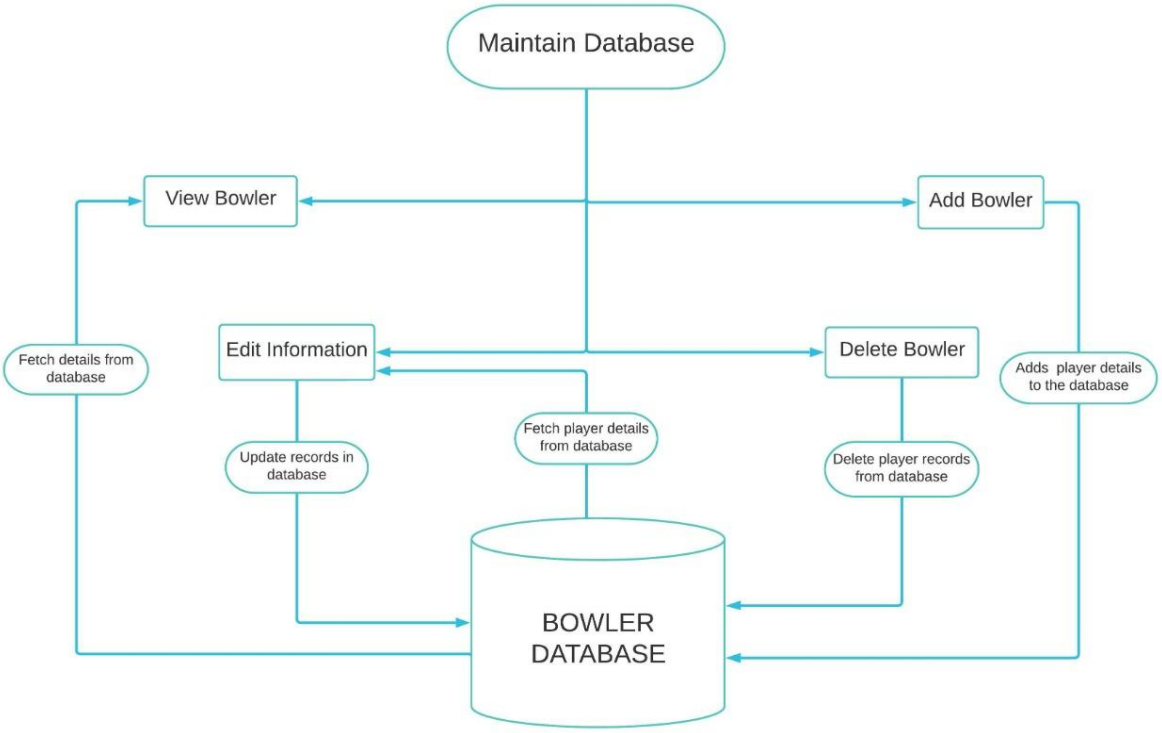
Sharing of knowledge and Multi-user Transaction Processing :

- A multi-user DBMS, as its name implies, must allow multiple users to access the database at an equivalent time or concurrently.

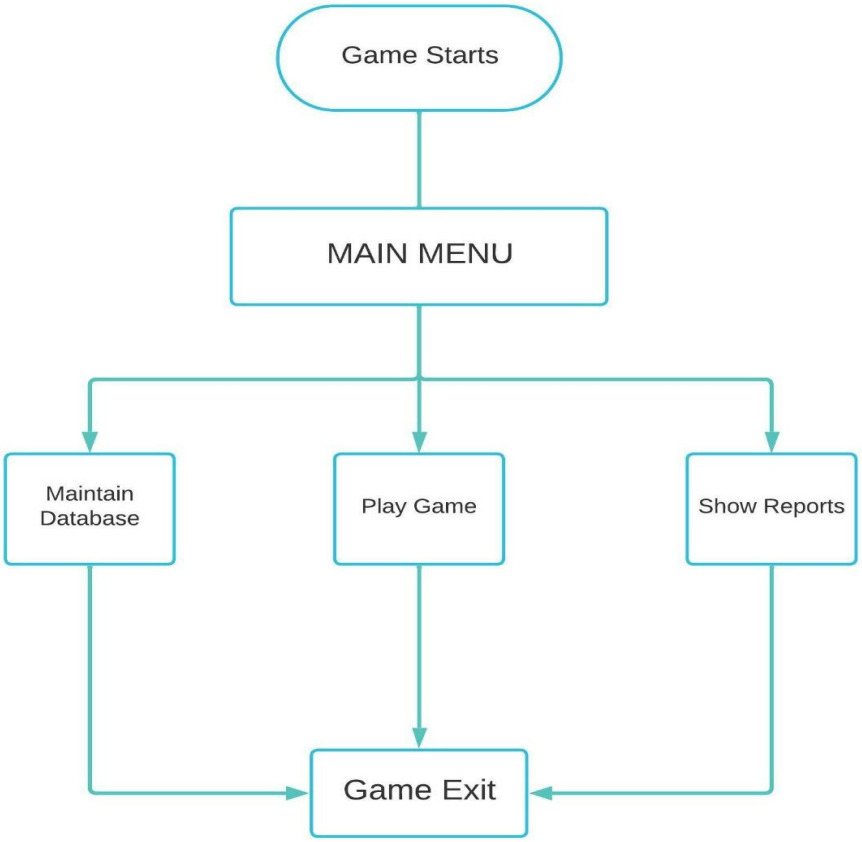
4. DETAILED SYSTEM DESIGN:



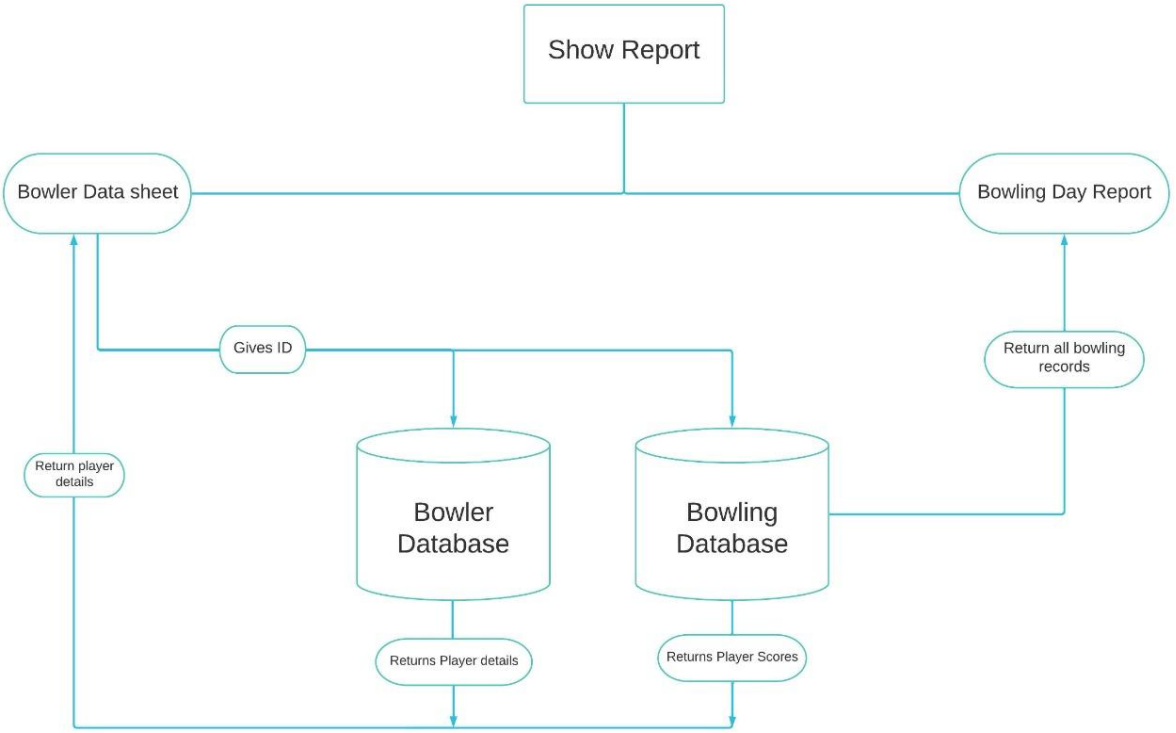
4.1 Flow Chart of the application



4.2 Flow Chart for Maintain database function



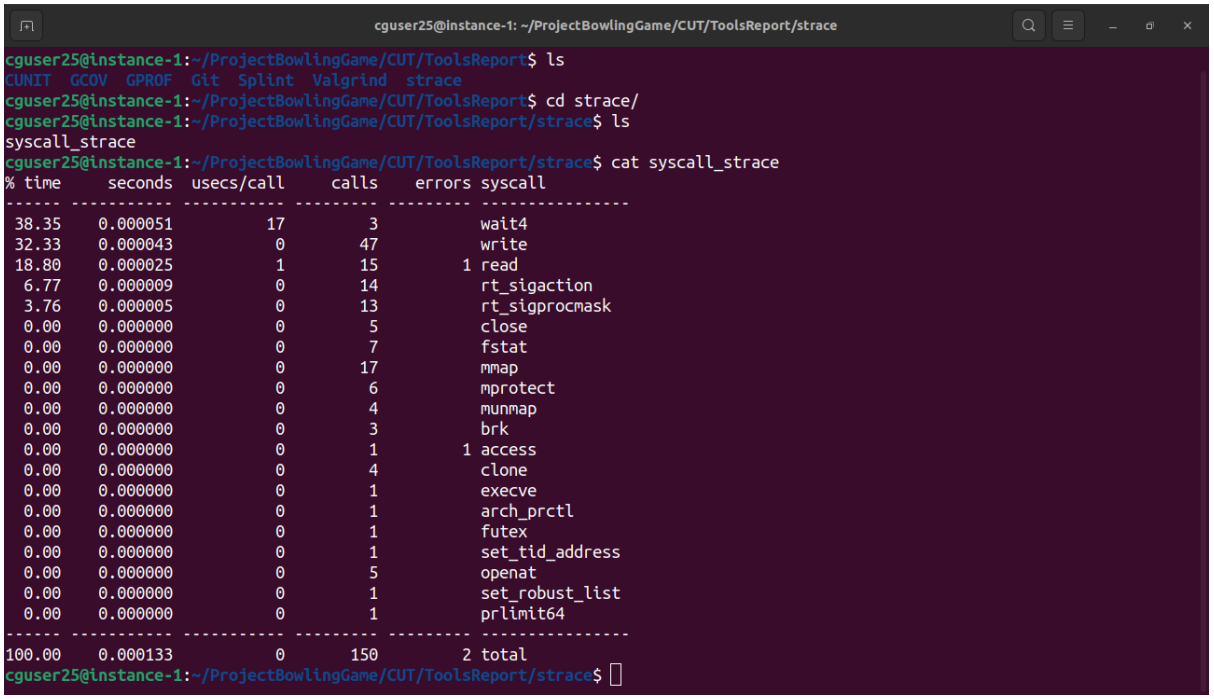
4.3 Flow chart for Main menu of the application



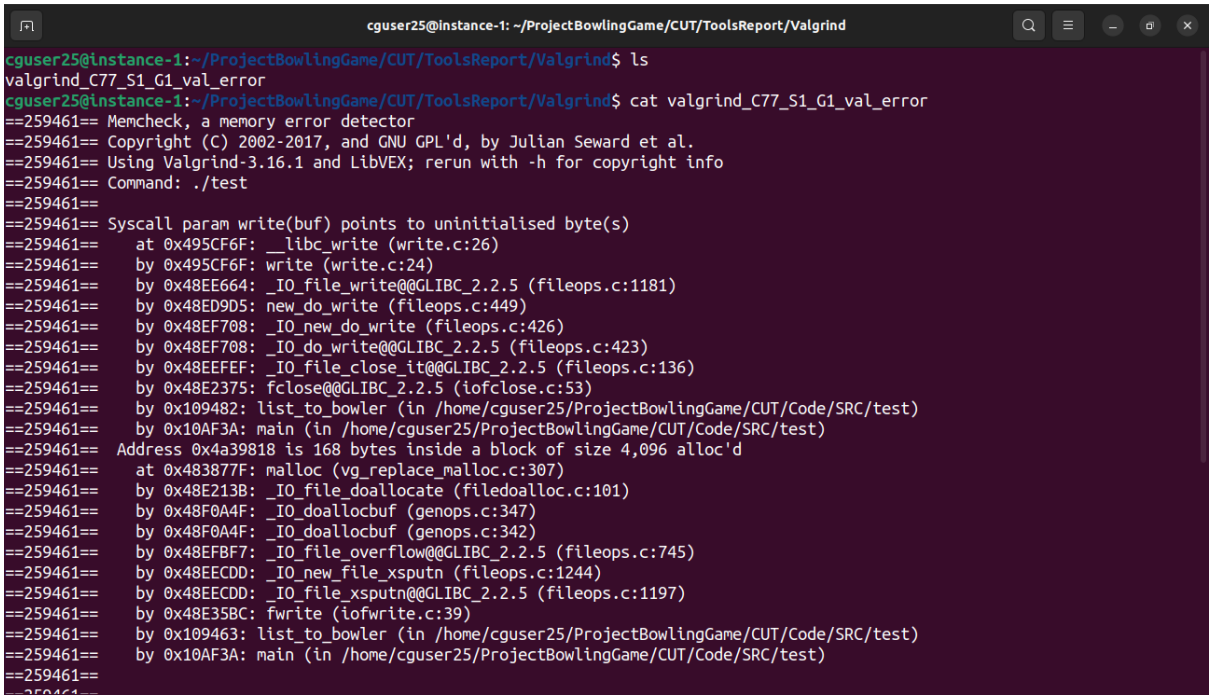
4.4 Flow chart for Show report of the application

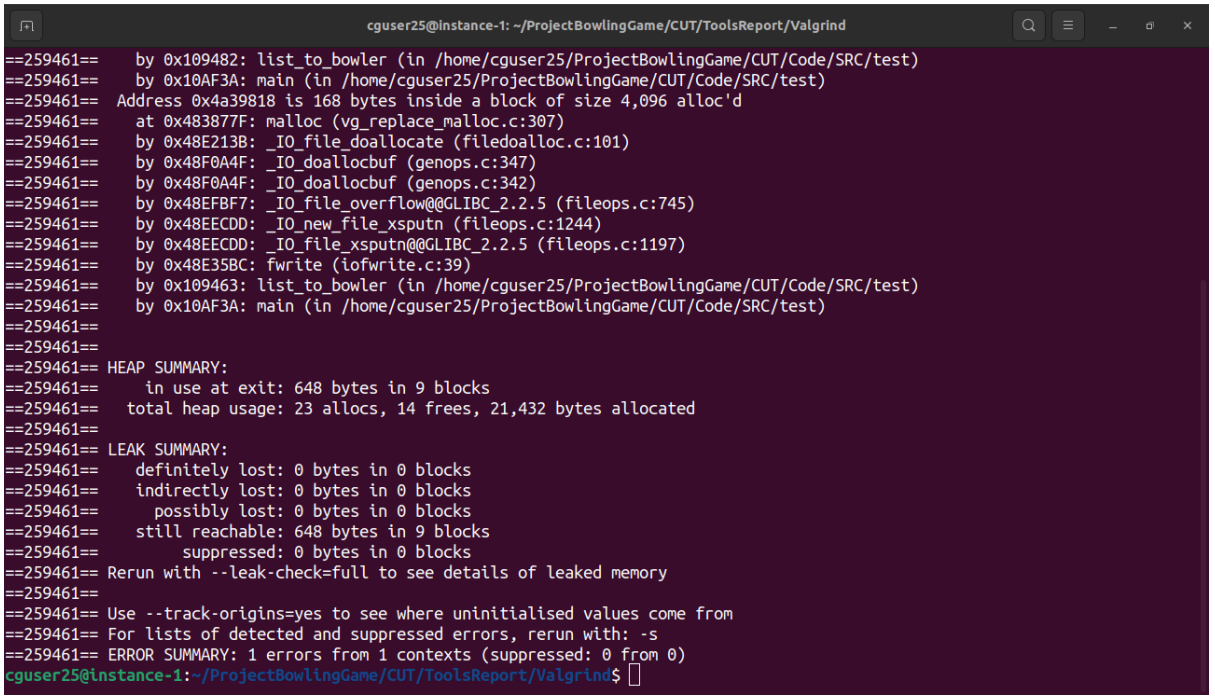
5. TOOLS REPORT

5.1 Strace

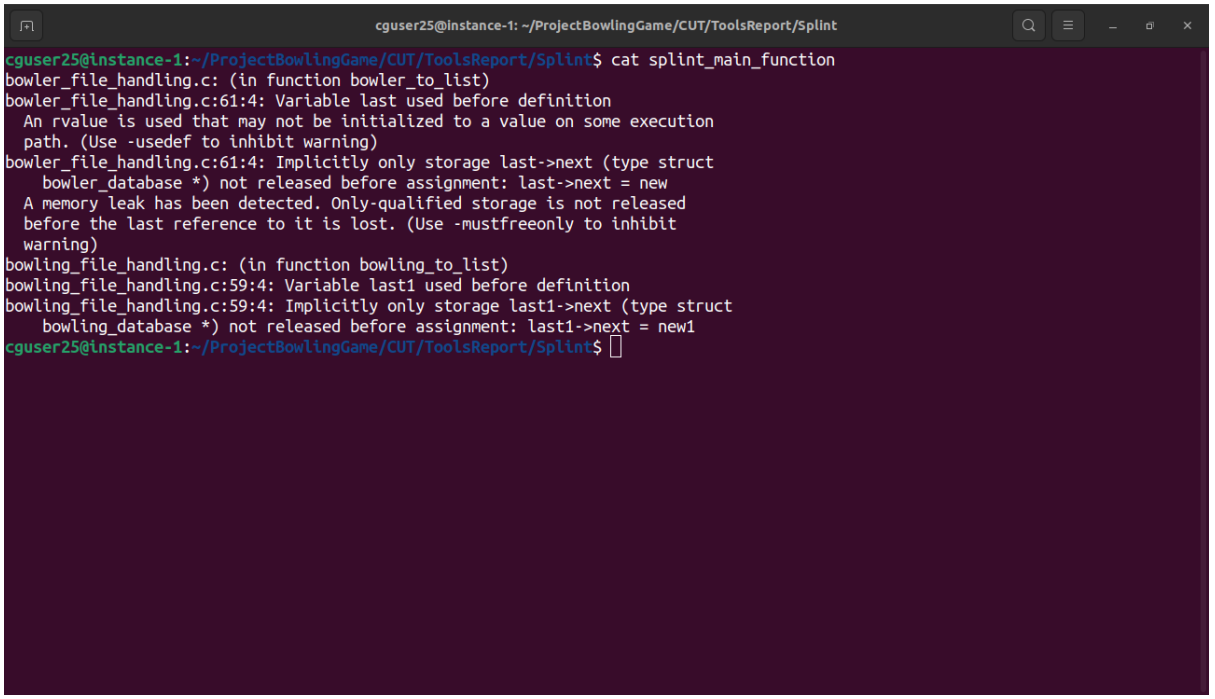


5.2 Valgrind




cguser25@instance-1: ~/ProjectBowlingGame/CUT/ToolsReport/Valgrind
==259461==   by 0x109482: list_to_bowler (in /home/cguser25/ProjectBowlingGame/CUT/Code/SRC/test)
==259461==   by 0x10AF3A: main (in /home/cguser25/ProjectBowlingGame/CUT/Code/SRC/test)
==259461== Address 0x4a39818 is 168 bytes inside a block of size 4,096 alloc'd
==259461==   at 0x483877F: malloc (vg_replace_malloc.c:307)
==259461==   by 0x48E213B: _IO_file_doallocate (filedoalloc.c:101)
==259461==   by 0x48F0A4F: _IO_doallocbuf (genops.c:347)
==259461==   by 0x48F0A4F: _IO_doallocbuf (genops.c:342)
==259461==   by 0x48EFBF7: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:745)
==259461==   by 0x48EECDD: _IO_new_file_xsputn (fileops.c:1244)
==259461==   by 0x48EECDD: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1197)
==259461==   by 0x48E35BC: fwrite (iofwrite.c:39)
==259461==   by 0x109463: list_to_bowler (in /home/cguser25/ProjectBowlingGame/CUT/Code/SRC/test)
==259461==   by 0x10AF3A: main (in /home/cguser25/ProjectBowlingGame/CUT/Code/SRC/test)
==259461==
==259461==
==259461== HEAP SUMMARY:
==259461==   in use at exit: 648 bytes in 9 blocks
==259461==   total heap usage: 23 allocs, 14 frees, 21,432 bytes allocated
==259461==
==259461== LEAK SUMMARY:
==259461==   definitely lost: 0 bytes in 0 blocks
==259461==   indirectly lost: 0 bytes in 0 blocks
==259461==   possibly lost: 0 bytes in 0 blocks
==259461==   still reachable: 648 bytes in 9 blocks
==259461==   suppressed: 0 bytes in 0 blocks
==259461== Rerun with --leak-check=full to see details of leaked memory
==259461==
==259461== Use --track-origins=yes to see where uninitialised values come from
==259461== For lists of detected and suppressed errors, rerun with: -s
==259461== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
cguser25@instance-1:~/ProjectBowlingGame/CUT/ToolsReport/Valgrind$

5.3 Splint


cguser25@instance-1: ~/ProjectBowlingGame/CUT/ToolsReport/Splint
cguser25@instance-1:~/ProjectBowlingGame/CUT/ToolsReport/Splint$ cat splint_main_function
bowler_file_handling.c: (in function bowler_to_list)
bowler_file_handling.c:61:4: Variable last used before definition
  An rvalue is used that may not be initialized to a value on some execution
  path. (Use -usedef to inhibit warning)
bowler_file_handling.c:61:4: Implicitly only storage last->next (type struct
    bowler_database *) not released before assignment: last->next = new
  A memory leak has been detected. Only-qualified storage is not released
  before the last reference to it is lost. (Use -mustfreeonly to inhibit
  warning)
bowling_file_handling.c: (in function bowling_to_list)
bowling_file_handling.c:59:4: Variable last1 used before definition
bowling_file_handling.c:59:4: Implicitly only storage last1->next (type struct
    bowling_database *) not released before assignment: last1->next = new1
cguser25@instance-1:~/ProjectBowlingGame/CUT/ToolsReport/Splint$

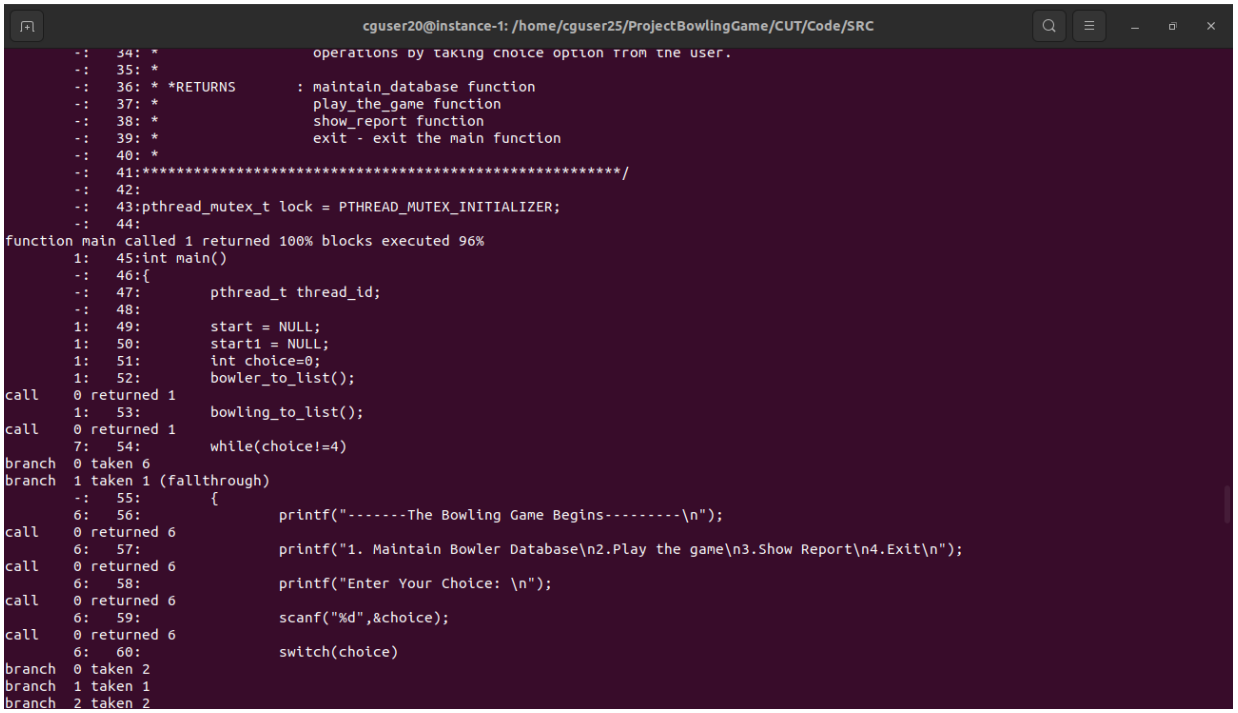
```
cguser25@instance-1: ~/ProjectBowlingGame/CUT/ToolsReport/Splint
cguser25@instance-1:~/ProjectBowlingGame/CUT/ToolsReport/Splint$ cat splint_play_function
play_the_game_function.c: (in function play_the_game_function)
play_the_game_function.c:268:5: Implicitly only storage new1->next (type struct
    bowling_database *) not released before assignment: new1->next = start1
  A memory leak has been detected. Only-qualified storage is not released
  before the last reference to it is lost. (Use -mustfreeonly to inhibit
  warning)
play_the_game_function.c:268:5: Unqualified storage start1 assigned to
                                implicitly only: new1->next = start1
  Unqualified storage is transferred in an inconsistent way. (Use
  -unqualifiedtrans to inhibit warning)
play_the_game_function.c:282:3: Clauses exit with new1 referencing fresh
    storage in true branch, kept storage in false branch
  The state of a variable is different depending on which branch is taken. This
  means no annotation can sensibly be applied to the storage. (Use -branchstate
  to inhibit warning)
   play_the_game_function.c:276:5: Storage new1 becomes kept
   play_the_game_function.c:220:15: Fresh storage new1 created
play_the_game_function.c:286:11: Function returns with global start1
                                 referencing kept storage
  A global variable does not satisfy its annotations when control is
  transferred. (Use -globstate to inhibit warning)
   play_the_game_function.c:277:5: Storage start1 becomes kept

cguser25@instance-1:~/ProjectBowlingGame/CUT/ToolsReport/Splint$
```

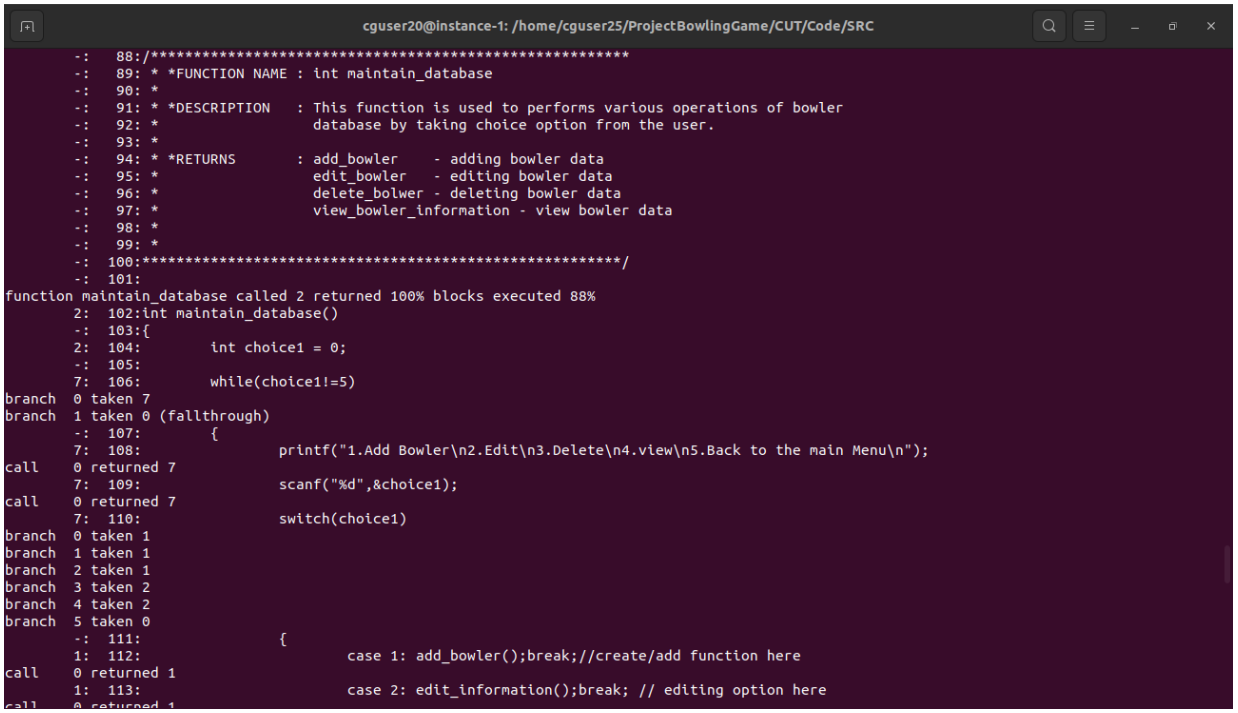
5.4 Gcov

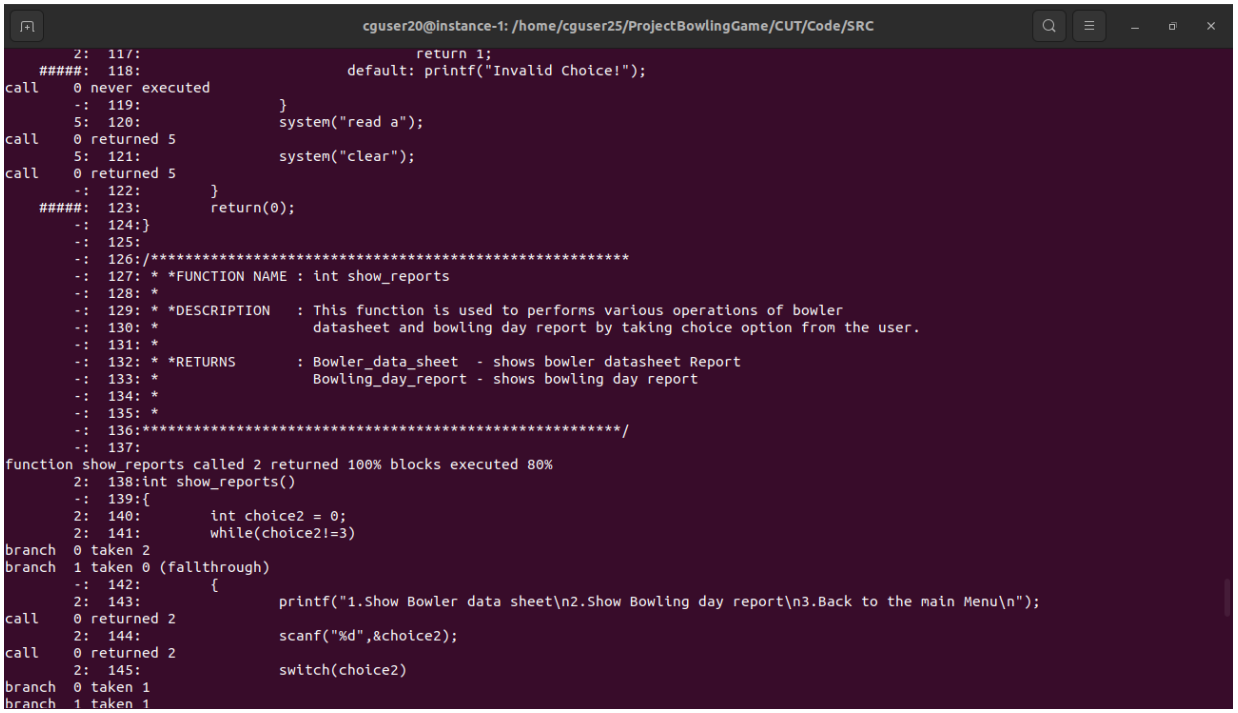
Main.c.gcov:

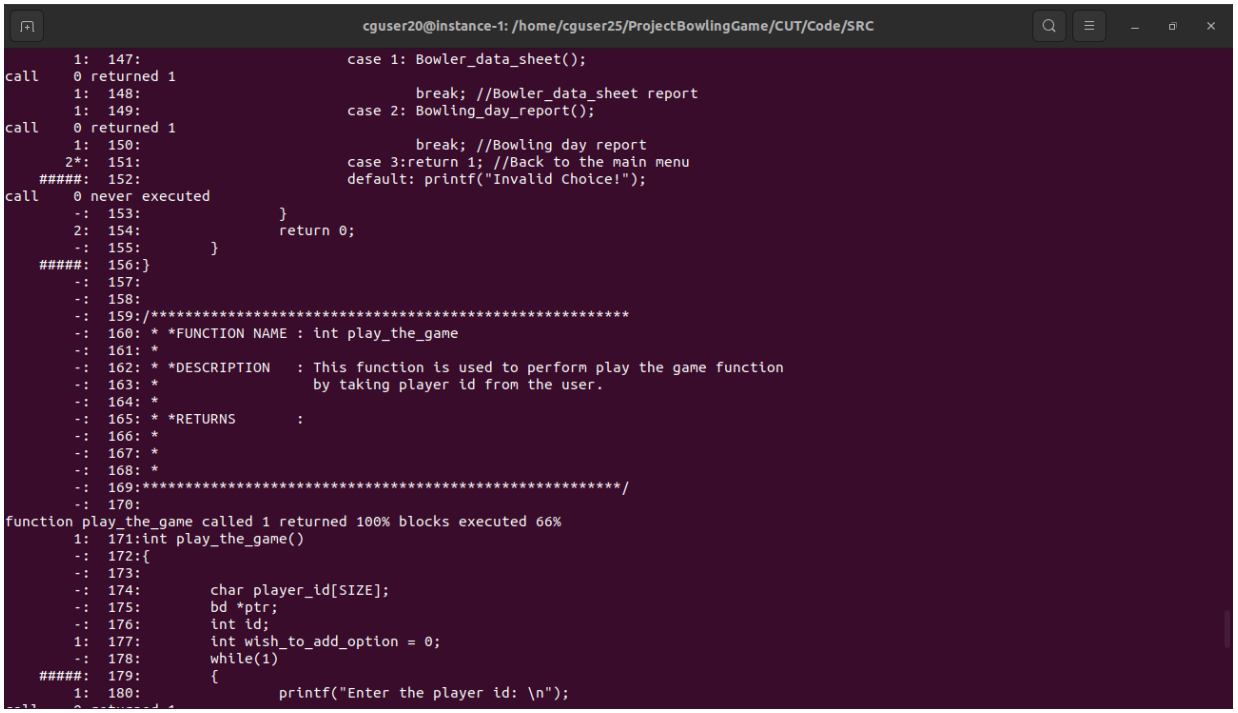
```
cguser20@instance-1:/home/cguser25/ProjectBowlingGame/CUT/Code/SRC$ cat main.c.gcov
        -:    0:Source:main.c
        -:    0:Graph:main.gcno
        -:    0:Data:main.gcda
        -:    0:Runs:1
        -:    1:/*******************************************
        -:    2: * *FILENAME          : main.c
        -:    3: *
        -:    4: * *DESCRITION        : This file defines the functions that consists of various subfunctions
        -:    5: *                      to perform certain opertions.
        -:    6: *
        -:    7: * Revision History   :
        -:    8: *
        -:    9: *      Date                    Name                    Reason
        -:   10: *
        -:   11: * 27th Aug 2022               ----                    -----
        -:   12: *
        -:   13: *
        -:   14:*******************************************/
        -:   15:
        -:   16:#include<stdio.h>
        -:   17:#include<stdlib.h>
        -:   18:#include "structure.h"
        -:   19:
        -:   20:#include "bowler_file_handling.c"
        -:   21:#include "bowling_file_handling.c"
        -:   22:#include "play_the_game_function.c"
        -:   23:#include "show_report.c"
        -:   24:#include "maintain_database_function.c"
        -:   25:#include <unistd.h>
        -:   26:#include <pthread.h>
        -:   27://bd *start;
        -:   28:
        -:   29:
        -:   30:/*****************************************************
        -:   31: * *FUNCTION NAME : int main
        -:   32: *
        -:   33: * *DESCRIPTION   : This function calls the other functions to performs various
```

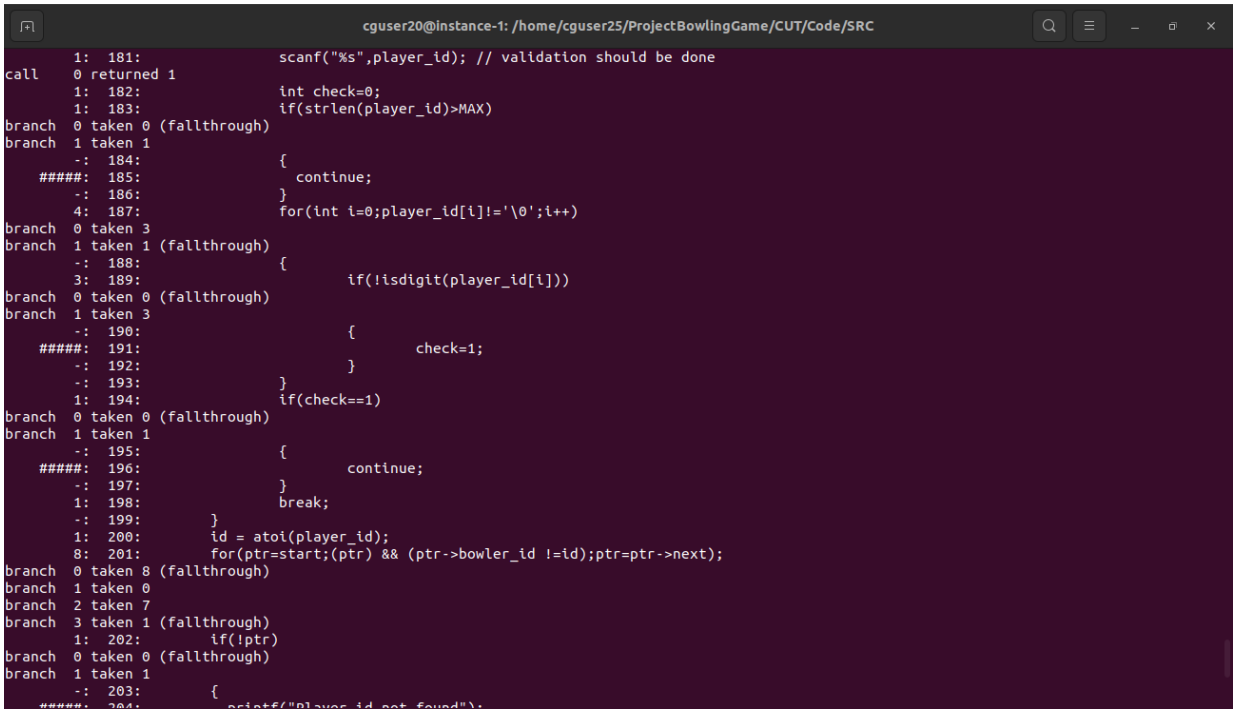

cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
-: 34: * operations by taking choice option from the user.
-: 35: *
-: 36: * *RETURNS : maintain_database function
-: 37: * play_the_game function
-: 38: * show_report function
-: 39: * exit - exit the main function
-: 40: *
-: 41:**/
-: 42:
-: 43:pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
-: 44:
function main called 1 returned 100% blocks executed 96%
1: 45:int main()
-: 46:{
-: 47: pthread_t thread_id;
-: 48:
1: 49: start = NULL;
1: 50: start1 = NULL;
1: 51: int choice=0;
1: 52: bowler_to_list();
call 0 returned 1
1: 53: bowling_to_list();
call 0 returned 1
7: 54: while(choice!=4)
branch 0 taken 6
branch 1 taken 1 (fallthrough)
-: 55: {
6: 56: printf("-------The Bowling Game Begins---------\n");
call 0 returned 6
6: 57: printf("1. Maintain Bowler Database\n2.Play the game\n3.Show Report\n4.Exit\n");
call 0 returned 6
6: 58: printf("Enter Your Choice: \n");
call 0 returned 6
6: 59: scanf("%d",&choice);
call 0 returned 6
6: 60: switch(choice)
branch 0 taken 2
branch 1 taken 1
branch 2 taken 2

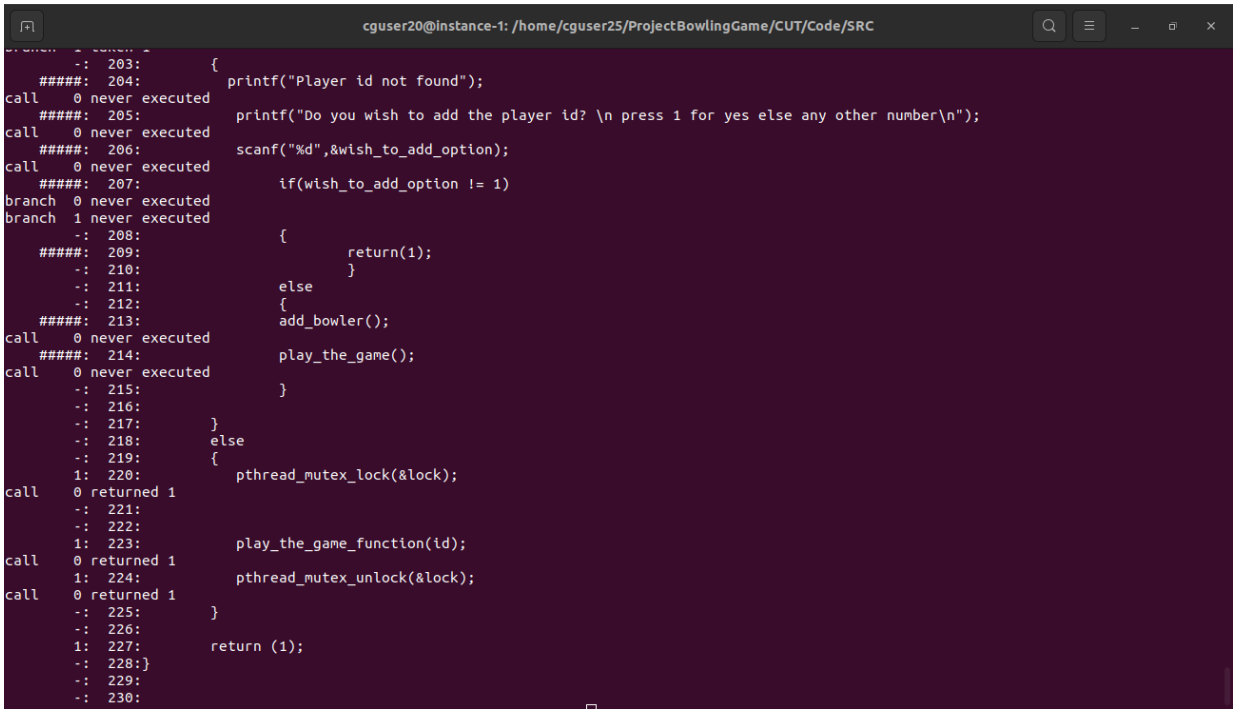

cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
branch 3 taken 1
branch 4 taken 0
-: 61: {
2: 62: case 1: maintain_database();break;
call 0 returned 2
1: 63: case 2: pthread_create(&thread_id,NULL,play_the_game,NULL);
call 0 returned 1
1: 64: pthread_join(thread_id,NULL);
call 0 returned 1
-: 65:
-: 66: // play_the_game();
-: 67:
1: 68: break; //play the game function here
-: 69:
2: 70: case 3:show_reports();
call 0 returned 2
2: 71: break;
1: 72: case 4:break;
#####: 73: default: printf("Invalid Choice!\n");
call 0 never executed
-: 74: }
-: 75: }
1: 76: if(start)
branch 0 taken 1 (fallthrough)
branch 1 taken 0
-: 77: {
1: 78: list_to_bowler();
call 0 returned 1
-: 79: }
1: 80: if(start1)
branch 0 taken 1 (fallthrough)
branch 1 taken 0
-: 81: {
1: 82: list_to_bowling();
call 0 returned 1
-: 83: }
-: 84: //pthread_mutex_destroy(&lock);
1: 85: return 0;


cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
-: 88:/***
-: 89: * *FUNCTION NAME : int maintain_database
-: 90: *
-: 91: * *DESCRIPTION : This function is used to performs various operations of bowler
-: 92: * database by taking choice option from the user.
-: 93: *
-: 94: * *RETURNS : add_bowler - adding bowler data
-: 95: * edit_bowler - editing bowler data
-: 96: * delete_bolwer - deleting bowler data
-: 97: * view_bowler_information - view bowler data
-: 98: *
-: 99: *
-: 100:***/
-: 101:
function maintain_database called 2 returned 100% blocks executed 88%
2: 102:int maintain_database()
-: 103:{
2: 104: int choice1 = 0;
-: 105:
7: 106: while(choice1!=5)
branch 0 taken 7
branch 1 taken 0 (fallthrough)
-: 107: {
7: 108: printf("1.Add Bowler\n2.Edit\n3.Delete\n4.view\n5.Back to the main Menu\n");
call 0 returned 7
7: 109: scanf("%d",&choice1);
call 0 returned 7
7: 110: switch(choice1)
branch 0 taken 1
branch 1 taken 1
branch 2 taken 1
branch 3 taken 2
branch 4 taken 2
branch 5 taken 0
-: 111: {
1: 112: case 1: add_bowler();break;//create/add function here
call 0 returned 1
1: 113: case 2: edit_information();break; // editing option here

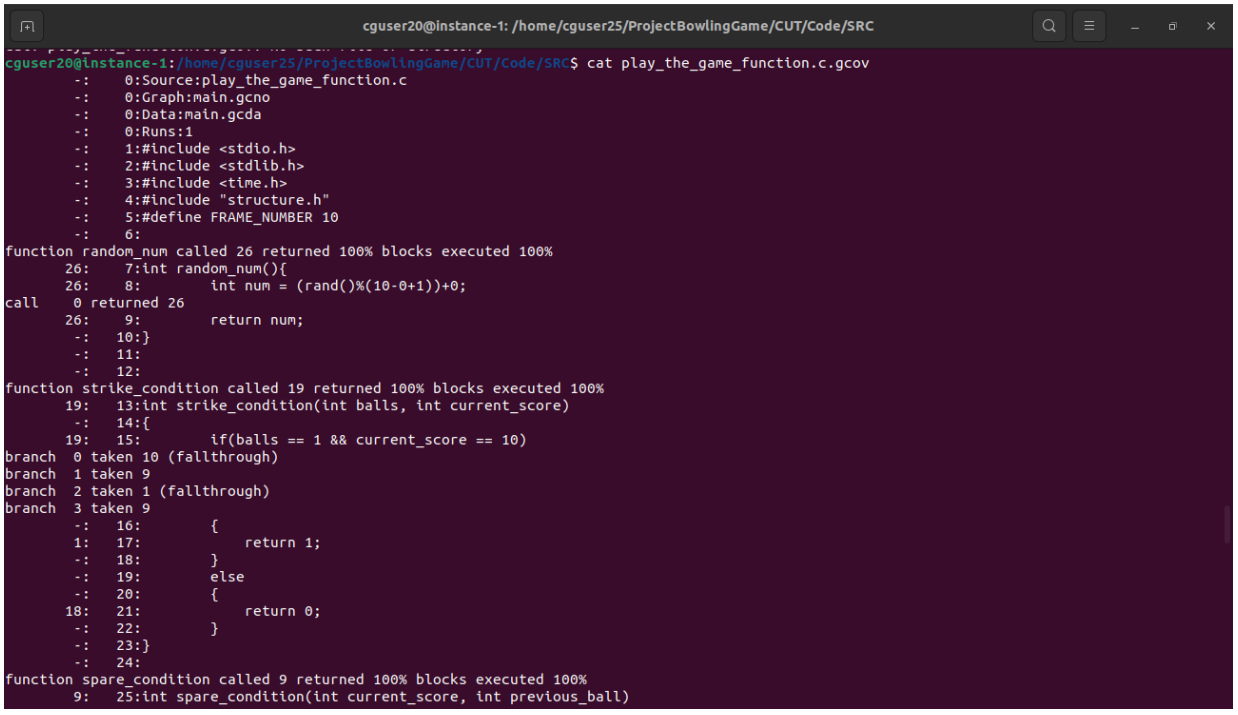

cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
2: 117: return 1;
#####: 118: default: printf("Invalid Choice!");
call 0 never executed
-: 119: }
5: 120: system("read a");
call 0 returned 5
5: 121: system("clear");
call 0 returned 5
-: 122: }
#####: 123: return(0);
-: 124:}
-: 125:
-: 126:/***
-: 127: * *FUNCTION NAME : int show_reports
-: 128: *
-: 129: * *DESCRIPTION : This function is used to performs various operations of bowler
-: 130: * datasheet and bowling day report by taking choice option from the user.
-: 131: *
-: 132: * *RETURNS : Bowler_data_sheet - shows bowler datasheet Report
-: 133: * Bowling_day_report - shows bowling day report
-: 134: *
-: 135: *
-: 136:***/
-: 137:
function show_reports called 2 returned 100% blocks executed 80%
2: 138:int show_reports()
-: 139:{
2: 140: int choice2 = 0;
2: 141: while(choice2!=3)
branch 0 taken 2
branch 1 taken 0 (fallthrough)
-: 142: {
2: 143: printf("1.Show Bowler data sheet\n2.Show Bowling day report\n3.Back to the main Menu\n");
call 0 returned 2
2: 144: scanf("%d",&choice2);
call 0 returned 2
2: 145: switch(choice2)
branch 0 taken 1

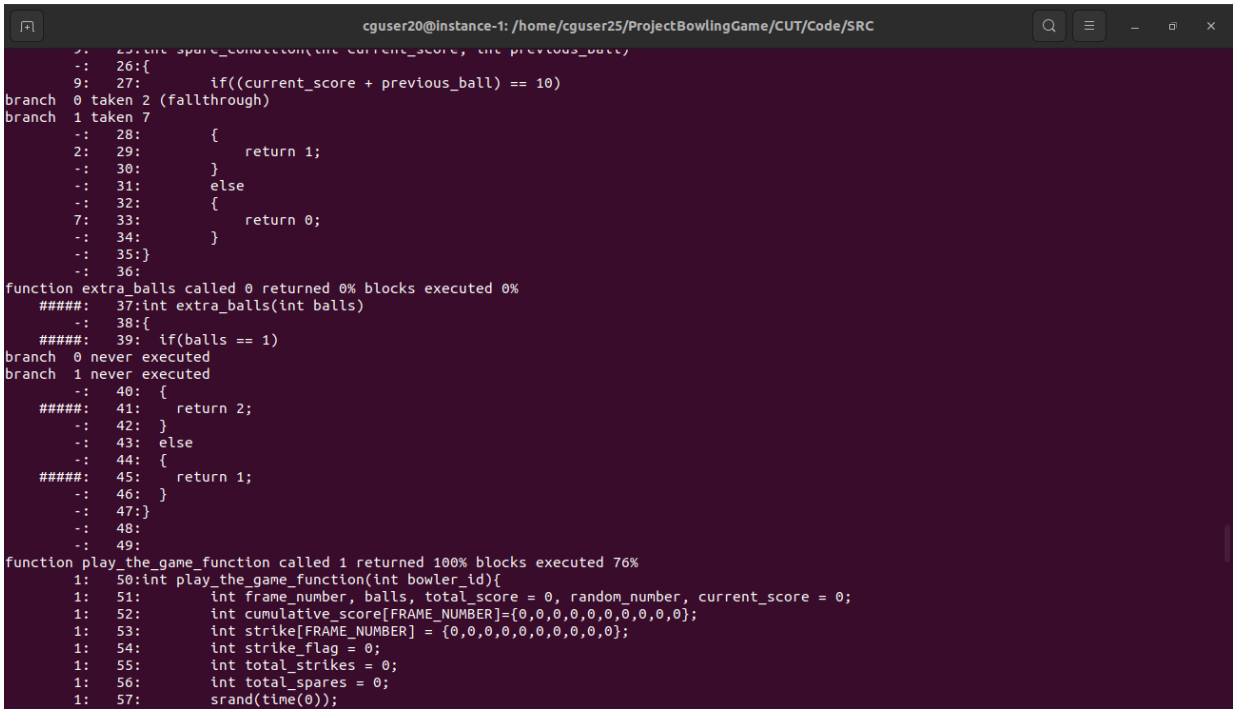

cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
1: 147: case 1: Bowler_data_sheet();
call 0 returned 1
1: 148: break; //Bowler_data_sheet report
1: 149: case 2: Bowling_day_report();
call 0 returned 1
1: 150: break; //Bowling day report
2*: 151: case 3:return 1; //Back to the main menu
#####: 152: default: printf("Invalid Choice!");
call 0 never executed
-: 153: }
2: 154: return 0;
-: 155: }
#####: 156:}
-: 157:
-: 158:
-: 159:/***
-: 160: * *FUNCTION NAME : int play_the_game
-: 161: *
-: 162: * *DESCRIPTION : This function is used to perform play the game function
-: 163: * by taking player id from the user.
-: 164: *
-: 165: * *RETURNS :
-: 166: *
-: 167: *
-: 168: *
-: 169:***/
-: 170:
function play_the_game called 1 returned 100% blocks executed 66%
1: 171:int play_the_game()
-: 172:{
-: 173:
-: 174: char player_id[SIZE];
-: 175: bd *ptr;
-: 176: int id;
1: 177: int wish_to_add_option = 0;
-: 178: while(1)
#####: 179: {
1: 180: printf("Enter the player id: \n");


cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
1: 181: scanf("%s",player_id); // validation should be done
call 0 returned 1
1: 182: int check=0;
1: 183: if(strlen(player_id)>MAX)
branch 0 taken 0 (fallthrough)
branch 1 taken 1
-: 184: {
#####: 185: continue;
-: 186: }
4: 187: for(int i=0;player_id[i]!='\0';i++)
branch 0 taken 3
branch 1 taken 1 (fallthrough)
-: 188: {
3: 189: if(!isdigit(player_id[i]))
branch 0 taken 0 (fallthrough)
branch 1 taken 3
-: 190: {
#####: 191: check=1;
-: 192: }
-: 193: }
1: 194: if(check==1)
branch 0 taken 0 (fallthrough)
branch 1 taken 1
-: 195: {
#####: 196: continue;
-: 197: }
1: 198: break;
-: 199: }
1: 200: id = atoi(player_id);
8: 201: for(ptr=start;(ptr) && (ptr->bowler_id !=id);ptr=ptr->next);
branch 0 taken 8 (fallthrough)
branch 1 taken 0
branch 2 taken 7
branch 3 taken 1 (fallthrough)
1: 202: if(!ptr)
branch 0 taken 0 (fallthrough)
branch 1 taken 1
-: 203: {



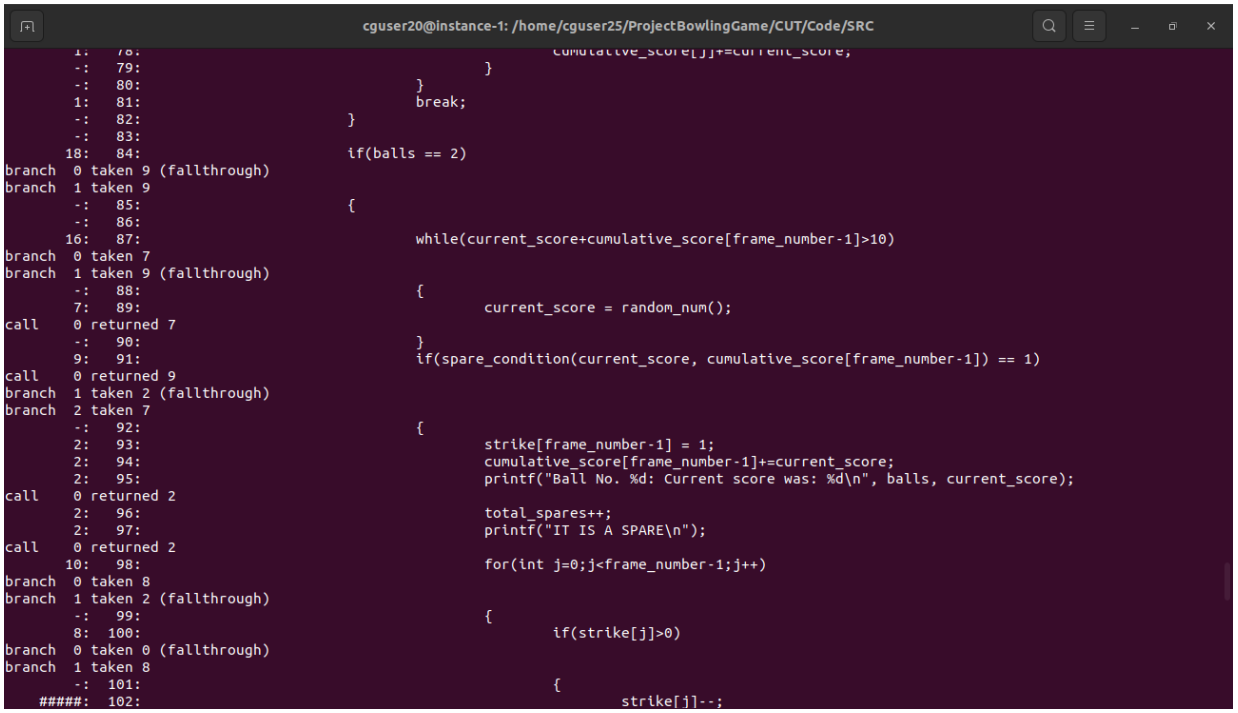
Play_the_function.c.gcov



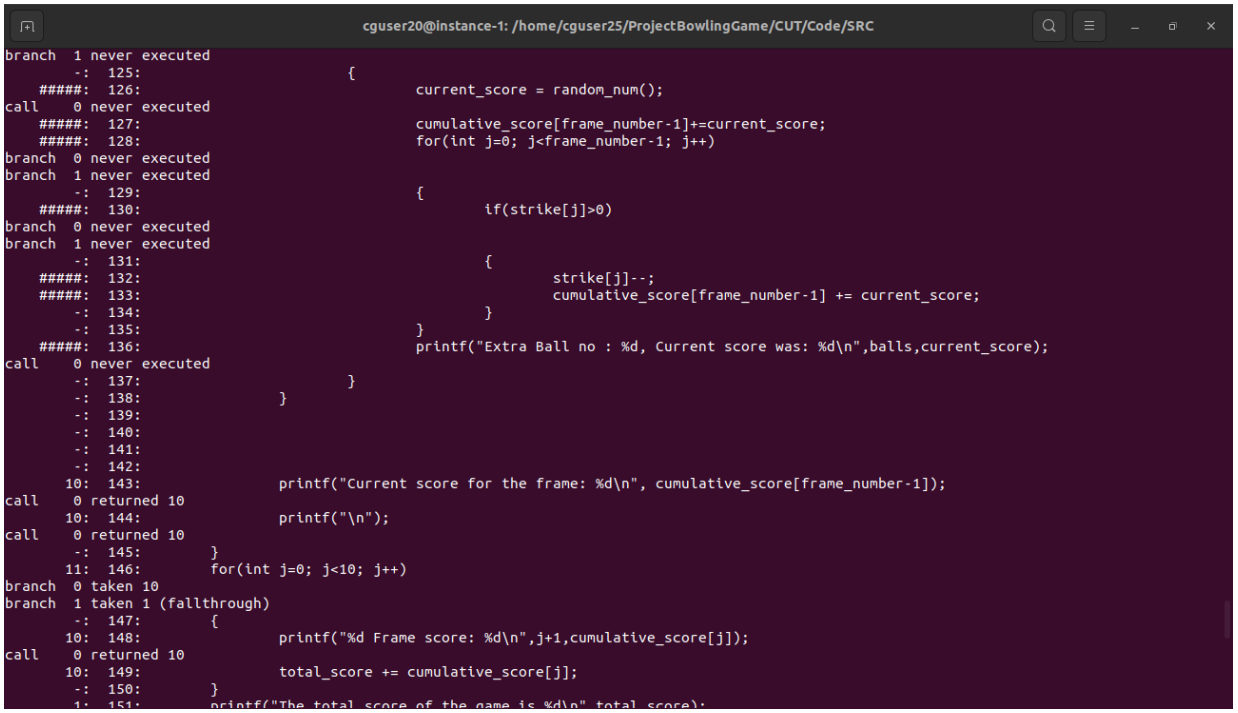


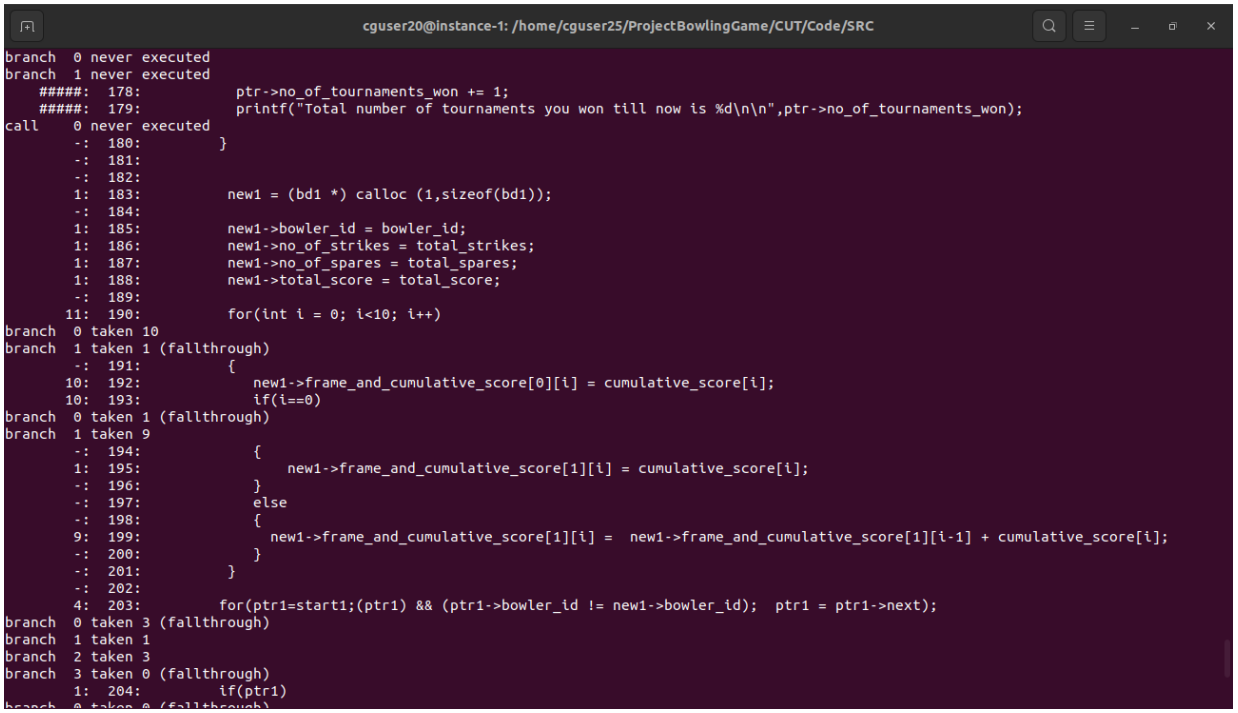

```
        -:   26:{
        9:   27:        if((current_score + previous_ball) == 10)
branch  0 taken 2 (fallthrough)
branch  1 taken 7
        -:   28:        {
        2:   29:            return 1;
        -:   30:        }
        -:   31:        else
        -:   32:        {
        7:   33:            return 0;
        -:   34:        }
        -:   35:}
        -:   36:
function extra_balls called 0 returned 0% blocks executed 0%
    #####:   37:int extra_balls(int balls)
        -:   38:{
    #####:   39:  if(balls == 1)
branch  0 never executed
branch  1 never executed
        -:   40:  {
    #####:   41:    return 2;
        -:   42:  }
        -:   43:  else
        -:   44:  {
    #####:   45:    return 1;
        -:   46:  }
        -:   47:}
        -:   48:
        -:   49:
function play_the_game_function called 1 returned 100% blocks executed 76%
        1:   50:int play_the_game_function(int bowler_id){
        1:   51:        int frame_number, balls, total_score = 0, random_number, current_score = 0;
        1:   52:        int cumulative_score[FRAME_NUMBER]={0,0,0,0,0,0,0,0,0,0};
        1:   53:        int strike[FRAME_NUMBER] = {0,0,0,0,0,0,0,0,0,0};
        1:   54:        int strike_flag = 0;
        1:   55:        int total_strikes = 0;
        1:   56:        int total_spares = 0;
        1:   57:        srand(time(0));
```

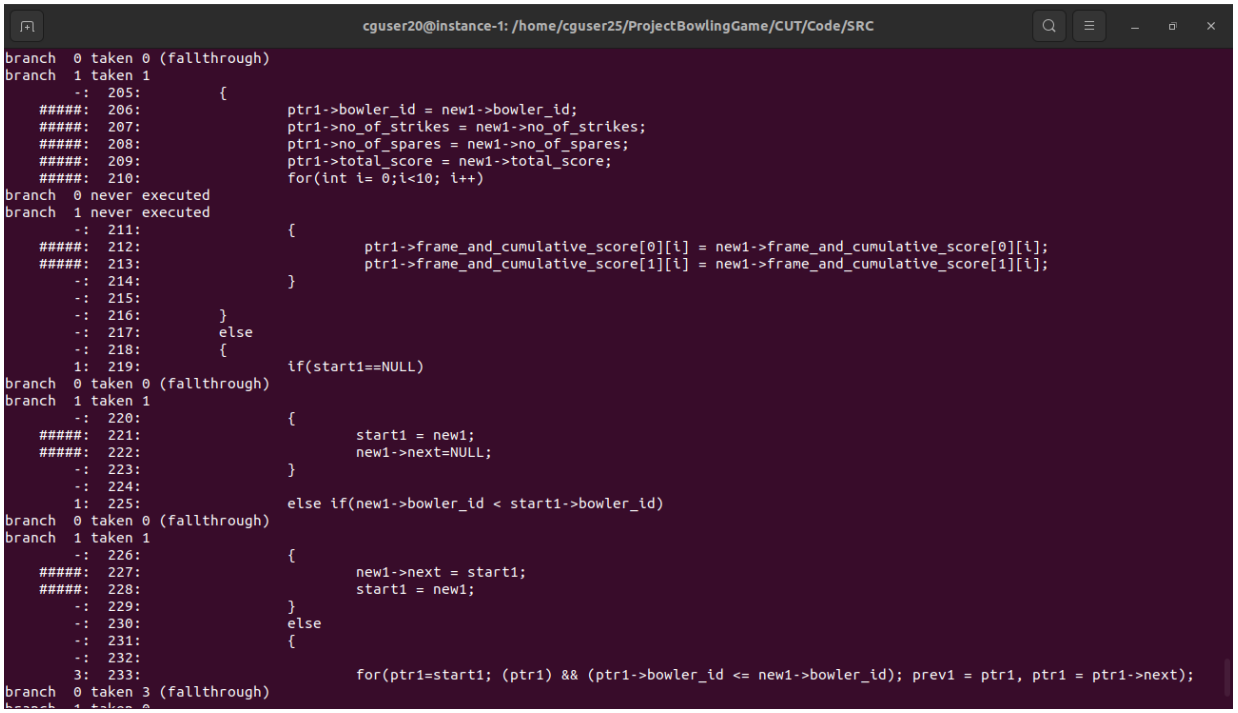

```
call    0 returned 1
call    1 returned 1
        1:   58:        int count=0;
        -:   59:
       11:   60:        for(frame_number = 1; frame_number<=10; frame_number++)
branch  0 taken 10
branch  1 taken 1 (fallthrough)
        -:   61:        {
       10:   62:                printf("***************** FRAME NUMBER: %d ******************\n",frame_number);
call    0 returned 10
       26:   63:                for(balls = 1; balls <= 2; balls++)
branch  0 taken 19
branch  1 taken 7 (fallthrough)
        -:   64:                {
       19:   65:                        current_score = random_num();
call    0 returned 19
       19:   66:                        if(strike_condition(balls,current_score) == 1)
call    0 returned 19
branch  1 taken 1 (fallthrough)
branch  2 taken 18
        -:   67:                        {
        1:   68:                                printf("Ball No. %d : Current score was: %d\n",balls, current_score);
call    0 returned 1
        1:   69:                                total_strikes++;
        1:   70:                                printf("IT IS A STRIKE\n");
call    0 returned 1
        1:   71:                                strike[frame_number-1] = 2;
        1:   72:                                cumulative_score[frame_number-1]+=current_score;
        2:   73:                                for(int j=0;j<frame_number-1;j++)
branch  0 taken 1
branch  1 taken 1 (fallthrough)
        -:   74:                                {
        1:   75:                                        if(strike[j]>0)
branch  0 taken 1 (fallthrough)
branch  1 taken 0
        -:   76:                                        {
        1:   77:                                                strike[j]--;
        1:   78:                                                cumulative_score[j]+=current_score;
```


cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
1: 78: cumulative_score[jj]+=current_score;
-: 79: }
-: 80: }
1: 81: break;
-: 82: }
-: 83:
18: 84: if(balls == 2)
branch 0 taken 9 (fallthrough)
branch 1 taken 9
-: 85: {
-: 86:
16: 87: while(current_score+cumulative_score[frame_number-1]>10)
branch 0 taken 7
branch 1 taken 9 (fallthrough)
-: 88: {
7: 89: current_score = random_num();
call 0 returned 7
-: 90: }
9: 91: if(spare_condition(current_score, cumulative_score[frame_number-1]) == 1)
call 0 returned 9
branch 1 taken 2 (fallthrough)
branch 2 taken 7
-: 92: {
2: 93: strike[frame_number-1] = 1;
2: 94: cumulative_score[frame_number-1]+=current_score;
2: 95: printf("Ball No. %d: Current score was: %d\n", balls, current_score);
call 0 returned 2
2: 96: total_spares++;
2: 97: printf("IT IS A SPARE\n");
call 0 returned 2
10: 98: for(int j=0;j<frame_number-1;j++)
branch 0 taken 8
branch 1 taken 2 (fallthrough)
-: 99: {
8: 100: if(strike[j]>0)
branch 0 taken 0 (fallthrough)
branch 1 taken 8
-: 101: {
#####: 102: strike[j]--;


cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
#####: 102: strike[jj]--;
#####: 103: cumulative_score[j]+=current_score;
-: 104: }
-: 105: }
2: 106: break;
-: 107: }
-: 108: }
96: 109: for(int j=0;j<frame_number-1;j++)
branch 0 taken 80
branch 1 taken 16 (fallthrough)
-: 110: {
80: 111: if(strike[j]>0)
branch 0 taken 3 (fallthrough)
branch 1 taken 77
-: 112: {
3: 113: strike[j]--;
3: 114: cumulative_score[j]+=current_score;
-: 115: }
-: 116: }
16: 117: printf("Ball No. %d: Current score was: %d\n", balls, current_score);
call 0 returned 16
-: 118:
16: 119: cumulative_score[frame_number-1] += current_score;
-: 120: }
10*: 121:if(frame_number == 10 && ((balls == 1 && current_score == 10) || (balls == 2 && cumulative_score[frame_number-1] == 10)))
branch 0 taken 1 (fallthrough)
branch 1 taken 9
branch 2 taken 0 (fallthrough)
branch 3 taken 1
branch 4 never executed
branch 5 never executed
branch 6 taken 0 (fallthrough)
branch 7 taken 1
branch 8 never executed
branch 9 never executed
-: 122: {
#####: 123: int extras = extra_balls(balls);
call 0 never executed
#####: 124: for(int balls=1; balls<=extras; balls++)

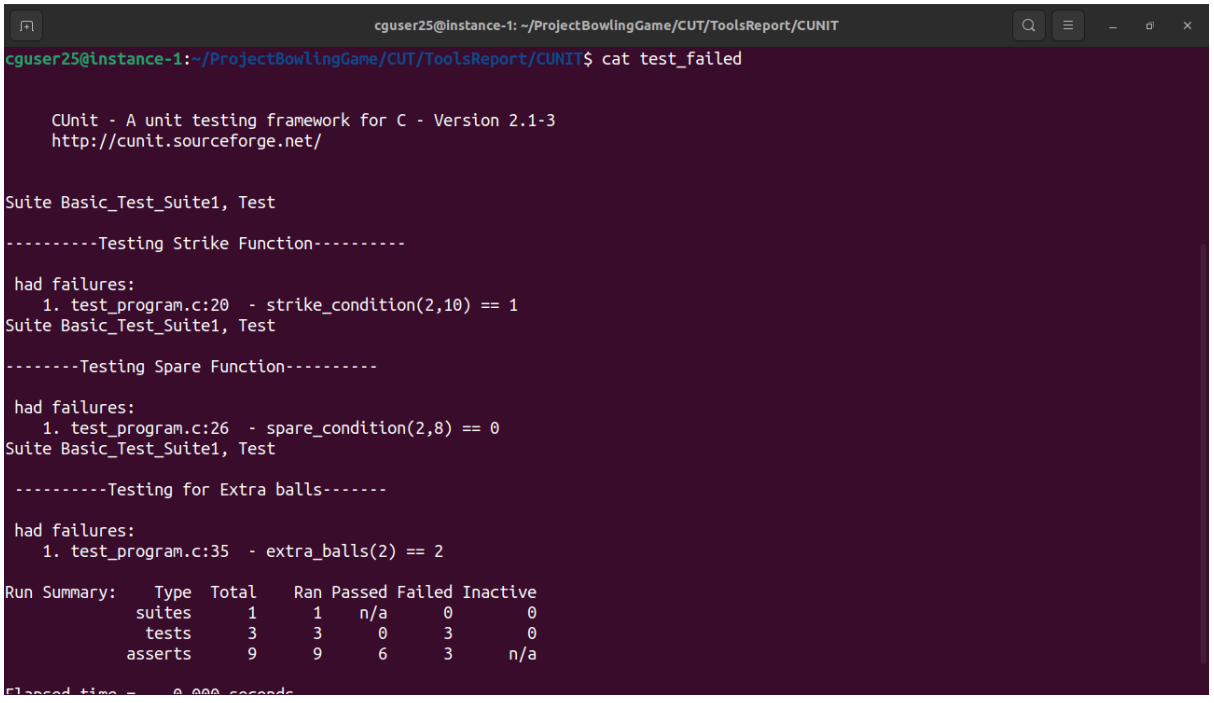

cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
branch 1 never executed
        -: 125:   {
    #####: 126:   current_score = random_num();
call   0 never executed
    #####: 127:   cumulative_score[frame_number-1]+=current_score;
    #####: 128:   for(int j=0; j<frame_number-1; j++)
branch 0 never executed
branch 1 never executed
        -: 129:   {
    #####: 130:   if(strike[j]>0)
branch 0 never executed
branch 1 never executed
        -: 131:   {
    #####: 132:   strike[j]--;
    #####: 133:   cumulative_score[frame_number-1] += current_score;
        -: 134:   }
        -: 135:   }
    #####: 136:   printf("Extra Ball no : %d, Current score was: %d\n",balls,current_score);
call   0 never executed
        -: 137:   }
        -: 138:   }
        -: 139:
        -: 140:
        -: 141:
        -: 142:
       10: 143:   printf("Current score for the frame: %d\n", cumulative_score[frame_number-1]);
call   0 returned 10
       10: 144:   printf("\n");
call   0 returned 10
        -: 145:   }
       11: 146:   for(int j=0; j<10; j++)
branch 0 taken 10
branch 1 taken 1 (fallthrough)
        -: 147:   {
       10: 148:   printf("%d Frame score: %d\n",j+1,cumulative_score[j]);
call   0 returned 10
       10: 149:   total_score += cumulative_score[j];
        -: 150:   }
        1: 151:   printf("The total score of the game is %d\n",total_score);
call   0 returned 1
        1: 152:   printf("Total strikes of the game is %d\n", total_strikes);
call   0 returned 1
        1: 153:   printf("Total spares of the game is %d\n", total_spares);
call   0 returned 1
        -: 154:
        -: 155:
        -: 156:
        -: 157:   // bowler_id, commu_array , no of strikes , no of spares, total score
        -: 158:
        -: 159:   bd1 *new1, *ptr1, *prev1;
        1: 160:   int flag_won = 1;
        2: 161:   for(ptr1=start1;(ptr1);ptr1 = ptr1->next)
branch 0 taken 2
branch 1 taken 0 (fallthrough)
        -: 162:   {
        2: 163:   if(total_score < ptr1->total_score)
branch 0 taken 1 (fallthrough)
branch 1 taken 1
        -: 164:   {
        1: 165:   printf("\nNot won the tournament\n");
call   0 returned 1
        1: 166:   flag_won = 0;
        1: 167:   break;
        -: 168:   }
        -: 169:
        -: 170:   }
        -: 171:
        1: 172:   if(flag_won == 1)
branch 0 taken 0 (fallthrough)
branch 1 taken 1
        -: 173:   {
    #####: 174:   printf("\n\n******CONGRATULATION YOU WON THE TOURNAMENT******\n\n");
call   0 never executed
        -: 175:
        -: 176:   bd *ptr;
    #####: 177:   for(ptr=start; (ptr->bowler_id != bowler_id); ptr=ptr->next);


cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
branch 0 never executed
branch 1 never executed
    #####: 178:   ptr->no_of_tournaments_won += 1;
    #####: 179:   printf("Total number of tournaments you won till now is %d\n\n",ptr->no_of_tournaments_won);
call   0 never executed
        -: 180:   }
        -: 181:
        -: 182:
        1: 183:   new1 = (bd1 *) calloc (1,sizeof(bd1));
        -: 184:
        1: 185:   new1->bowler_id = bowler_id;
        1: 186:   new1->no_of_strikes = total_strikes;
        1: 187:   new1->no_of_spares = total_spares;
        1: 188:   new1->total_score = total_score;
        -: 189:
       11: 190:   for(int i = 0; i<10; i++)
branch 0 taken 10
branch 1 taken 1 (fallthrough)
        -: 191:   {
       10: 192:   new1->frame_and_cumulative_score[0][i] = cumulative_score[i];
       10: 193:   if(i==0)
branch 0 taken 1 (fallthrough)
branch 1 taken 9
        -: 194:   {
        1: 195:   new1->frame_and_cumulative_score[1][i] = cumulative_score[i];
        -: 196:   }
        -: 197:   else
        -: 198:   {
        9: 199:   new1->frame_and_cumulative_score[1][i] = new1->frame_and_cumulative_score[1][i-1] + cumulative_score[i];
        -: 200:   }
        -: 201:   }
        -: 202:
        4: 203:   for(ptr1=start1;(ptr1) && (ptr1->bowler_id != new1->bowler_id); ptr1 = ptr1->next);
branch 0 taken 3 (fallthrough)
branch 1 taken 1
branch 2 taken 3
branch 3 taken 0 (fallthrough)
        1: 204:   if(ptr1)


cguser20@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
branch 0 taken 0 (fallthrough)
branch 1 taken 1
        -: 205:   {
    #####: 206:   ptr1->bowler_id = new1->bowler_id;
    #####: 207:   ptr1->no_of_strikes = new1->no_of_strikes;
    #####: 208:   ptr1->no_of_spares = new1->no_of_spares;
    #####: 209:   ptr1->total_score = new1->total_score;
    #####: 210:   for(int i= 0;i<10; i++)
branch 0 never executed
branch 1 never executed
        -: 211:   {
    #####: 212:   ptr1->frame_and_cumulative_score[0][i] = new1->frame_and_cumulative_score[0][i];
    #####: 213:   ptr1->frame_and_cumulative_score[1][i] = new1->frame_and_cumulative_score[1][i];
        -: 214:   }
        -: 215:
        -: 216:   }
        -: 217:   else
        -: 218:   {
        1: 219:   if(start1==NULL)
branch 0 taken 0 (fallthrough)
branch 1 taken 1
        -: 220:   {
    #####: 221:   start1 = new1;
    #####: 222:   new1->next=NULL;
        -: 223:   }
        -: 224:
        1: 225:   else if(new1->bowler_id < start1->bowler_id)
branch 0 taken 0 (fallthrough)
branch 1 taken 1
        -: 226:   {
    #####: 227:   new1->next = start1;
    #####: 228:   start1 = new1;
        -: 229:   }
        -: 230:   else
        -: 231:   {
        -: 232:
        3: 233:   for(ptr1=start1; (ptr1) && (ptr1->bowler_id <= new1->bowler_id); prev1 = ptr1, ptr1 = ptr1->next);
branch 0 taken 3 (fallthrough)

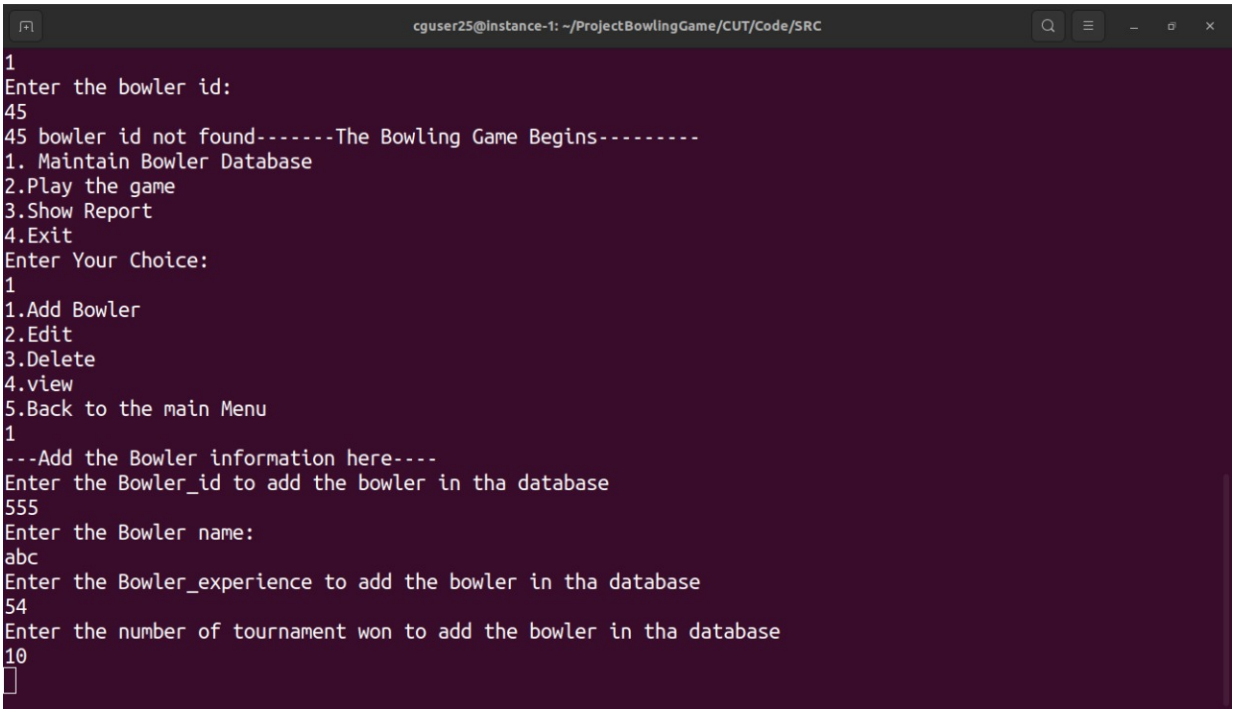
6. Testing

6.1 Unit Testing

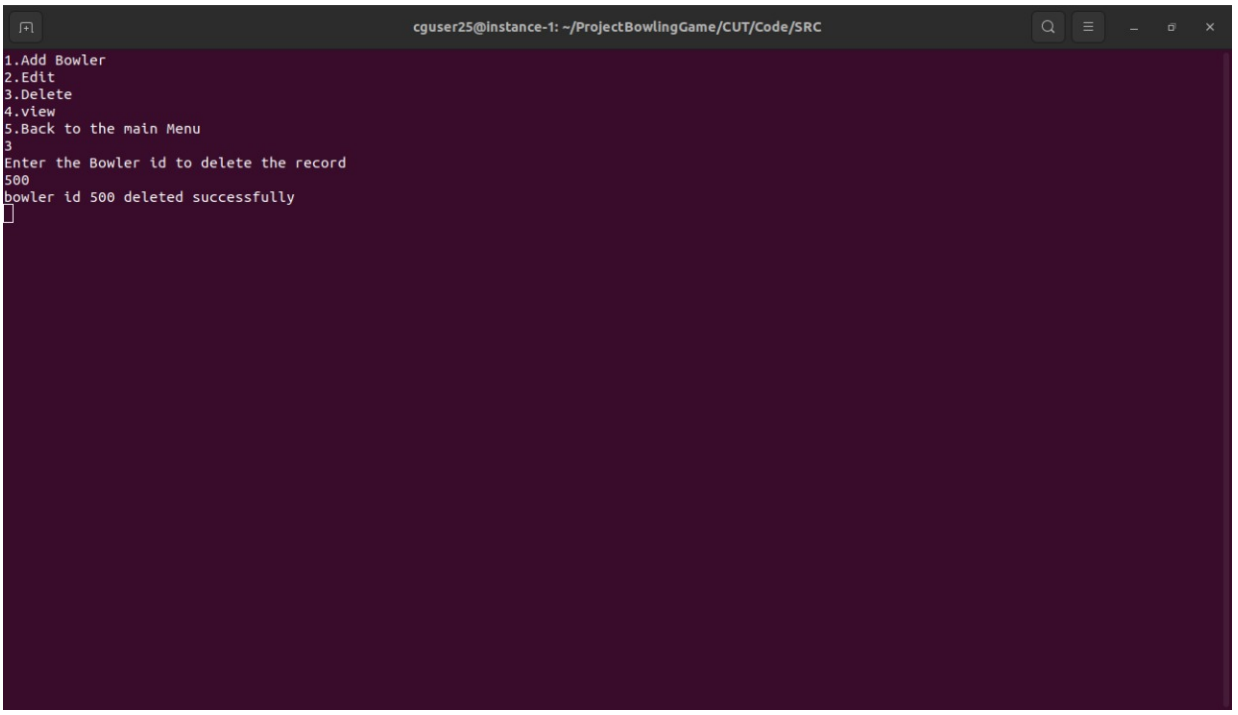


6.2 Integration Testing

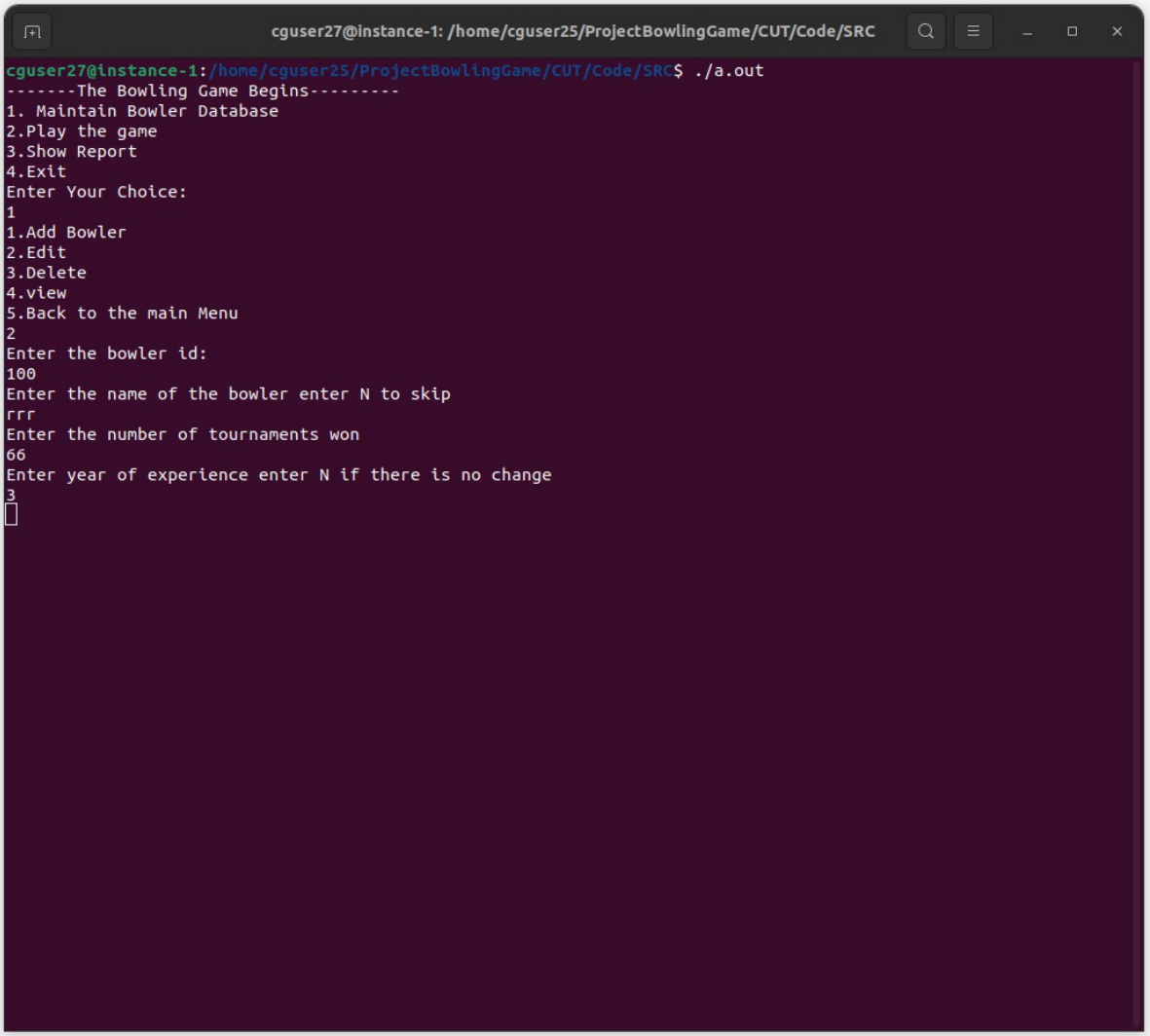
6.2.1 Add



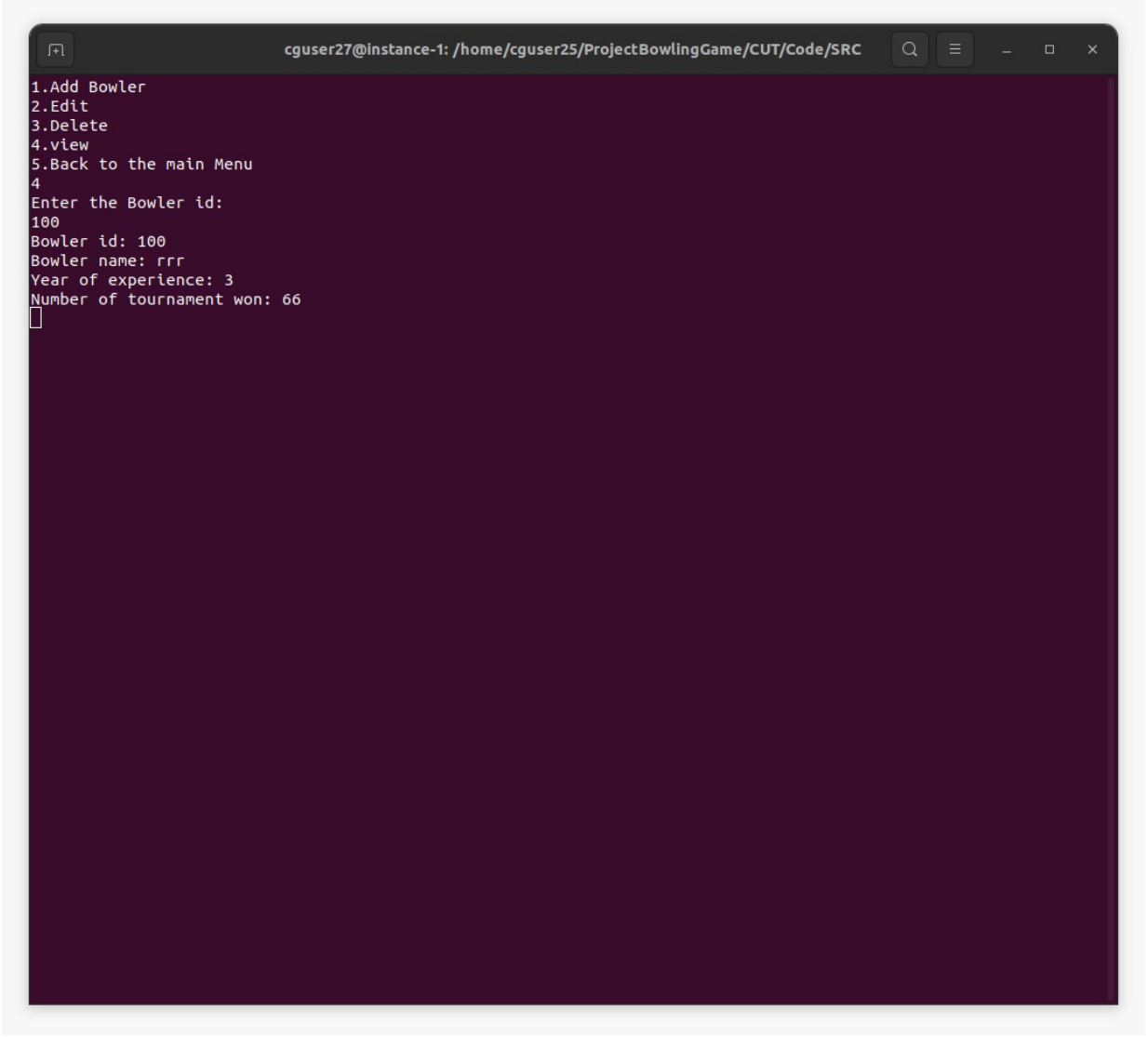
6.2.2 Delete



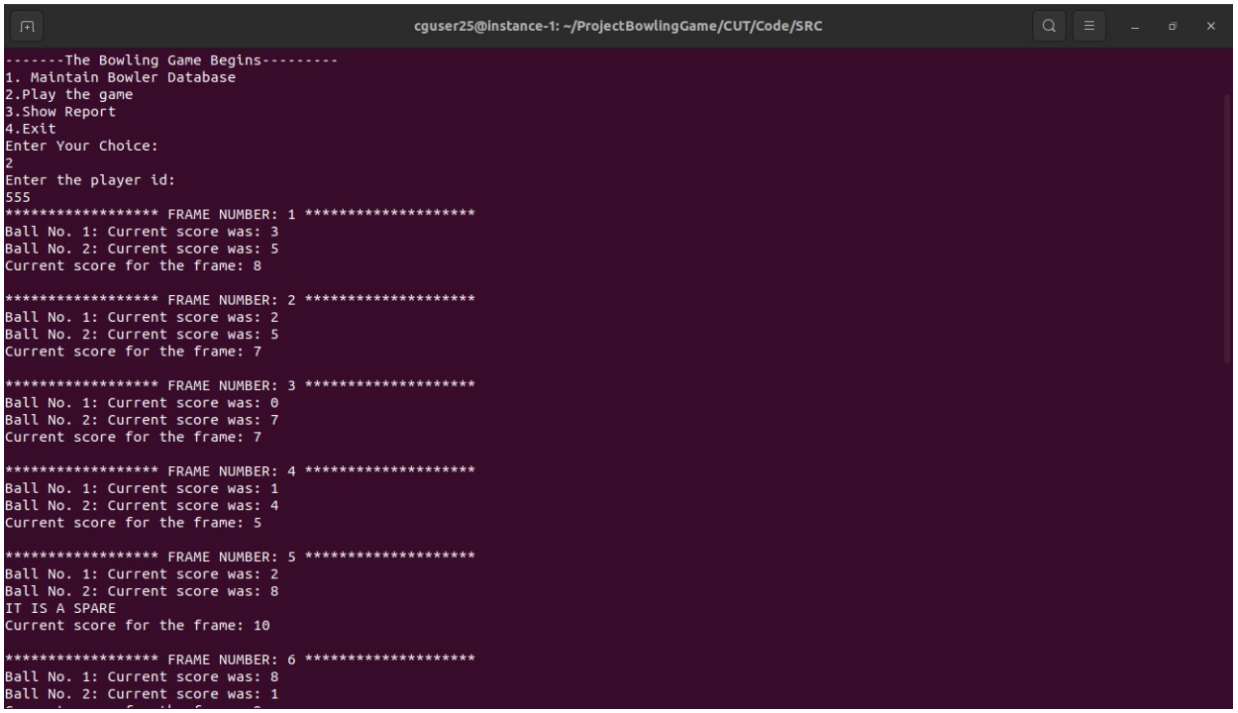
6.2.3 Edit

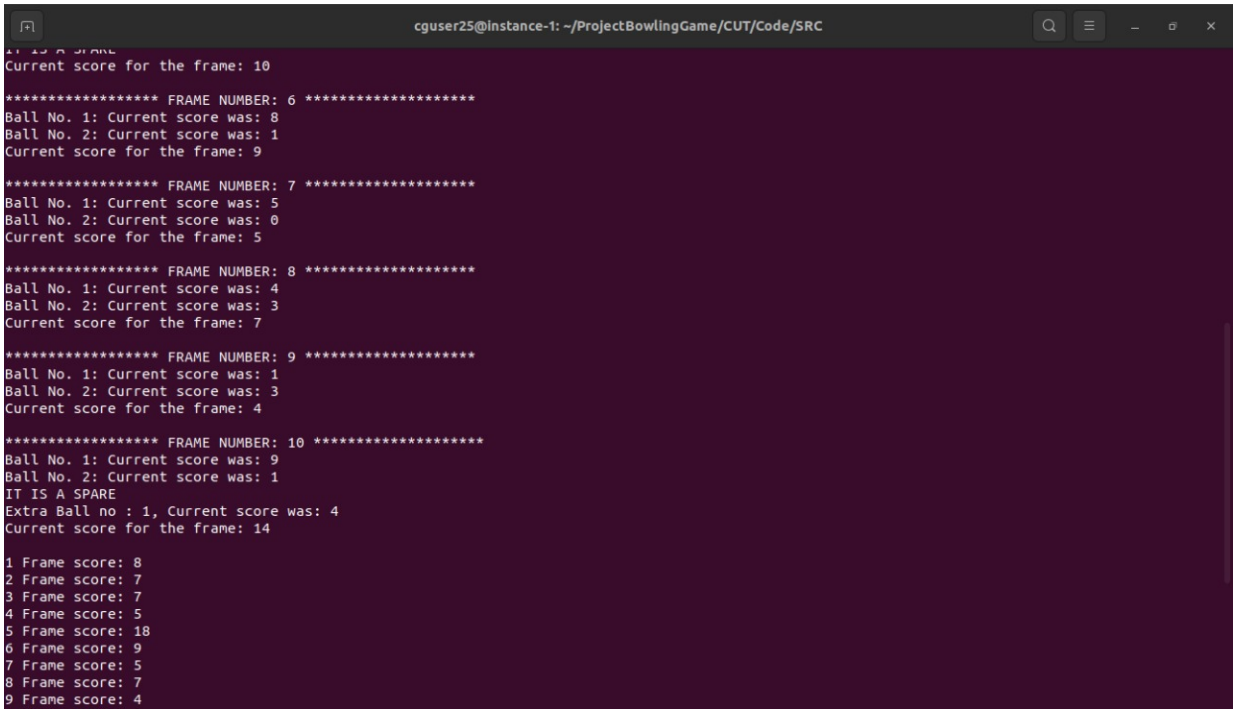

cguser27@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
cguser27@instance-1:/home/cguser25/ProjectBowlingGame/CUT/Code/SRC$ ./a.out
-------The Bowling Game Begins---------
1. Maintain Bowler Database
2.Play the game
3.Show Report
4.Exit
Enter Your Choice:
1
1.Add Bowler
2.Edit
3.Delete
4.view
5.Back to the main Menu
2
Enter the bowler id:
100
Enter the name of the bowler enter N to skip
rrr
Enter the number of tournaments won
66
Enter year of experience enter N if there is no change
3

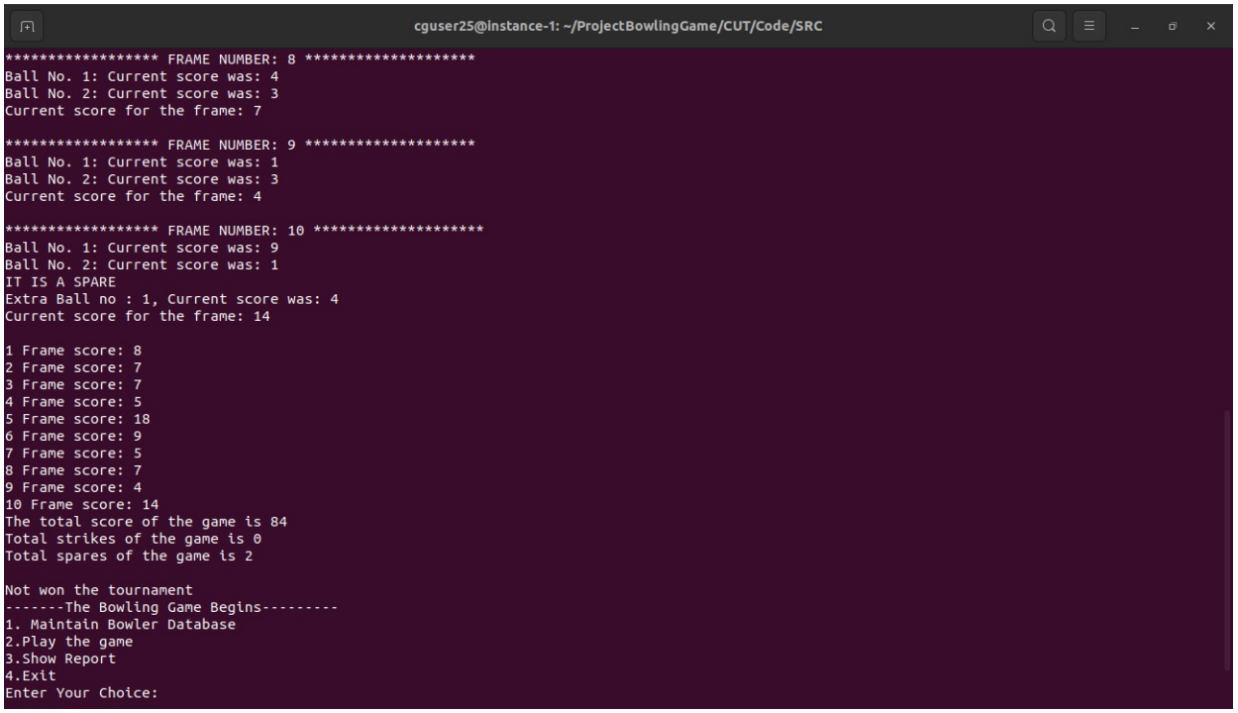
6.2.4 View


cguser27@instance-1: /home/cguser25/ProjectBowlingGame/CUT/Code/SRC
1.Add Bowler
2.Edit
3.Delete
4.view
5.Back to the main Menu
4
Enter the Bowler id:
100
Bowler id: 100
Bowler name: rrr
Year of experience: 3
Number of tournament won: 66

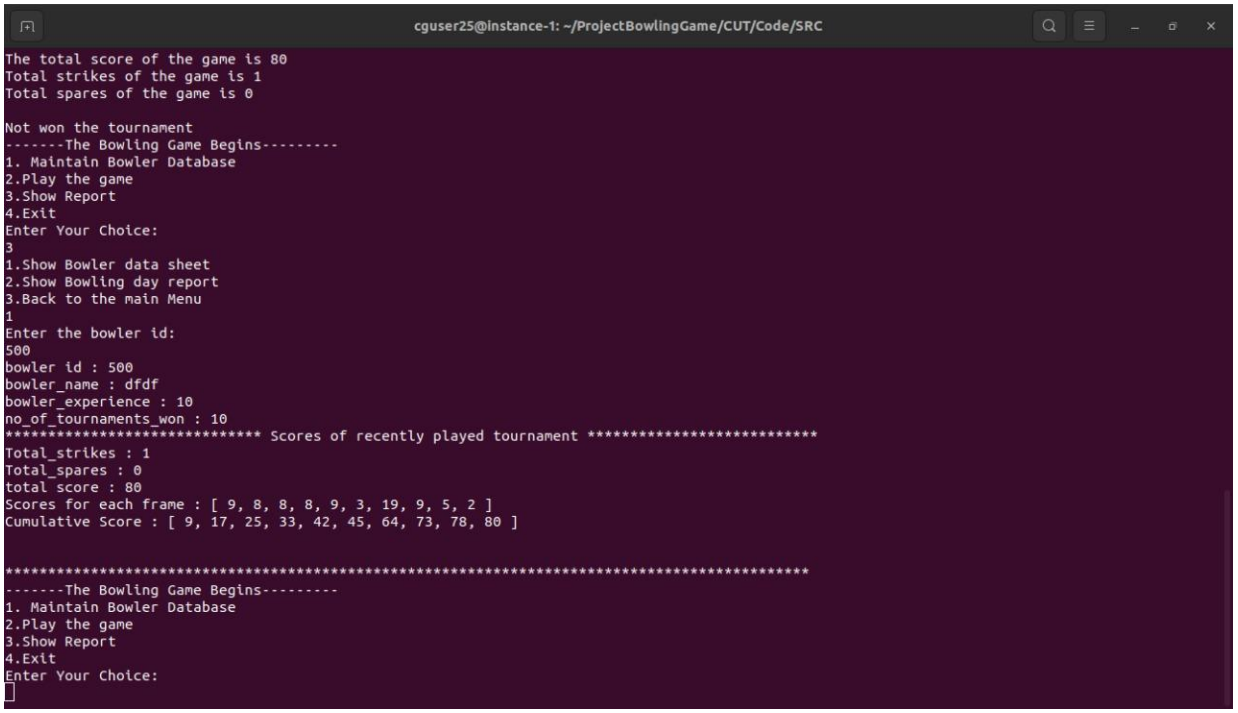
6.2.5 Play the game



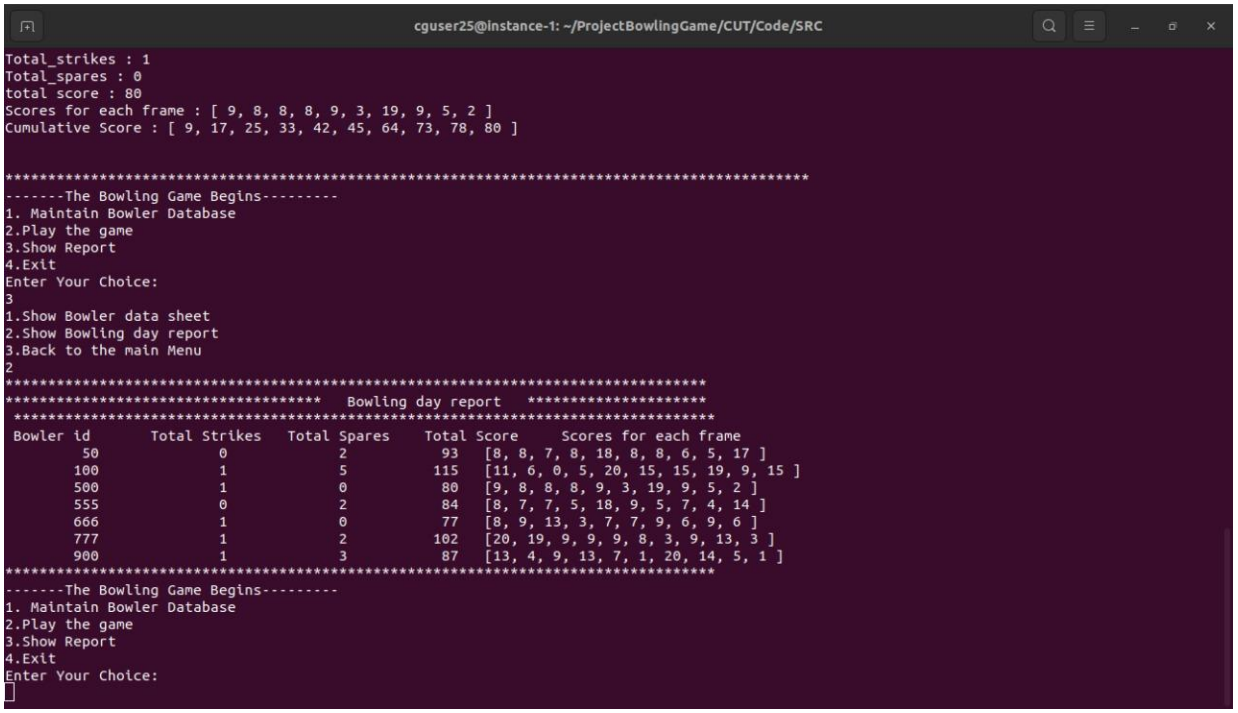




6.2.6 Bowler data sheet



6.2.7 Bowler day report



7. Requirements Traceability Matrix(RTM)

Req	Design Mapping	Code Mapping	UT Mapping	IT Mapping
BG_01	3.1.1	add_bowler()		IT_CASE1
BG_02	3.1.2	edit_information()		IT_CASE2
BG_03	3.1.3	delete_bowler()		IT_CASE3
BG_04	3.1.4	view_bowler_information()		IT_CASE4
BG_06	3.1.6	strike_condition()	Test_case 1	IT_CASE5
BG_07	3.1.7	extra_balls()	Test_case 3	IT_CASE5
BG_11	3.1.11	spare condition()	Test_case 2	IT_CASE5
BG_13	3.1.13	play_the_game_function()		IT_CASE5
BG_14	3.1.14	Bowler_data_sheet()		IT_CASE6
BG_15	3.1.15	Bowling_day_report()		IT_CASE7
BG_16	3.1.16	maintain_database()		IT_CASE1
BG_17	3.1.17	play_the_game()		IT_CASE5
BG_18	3.1.18	show_reports()		IT_CASE6