For this assignment we will be adding to / modifying an existing application.

In the **Bowling_Frame** class modify **update_scores** method so that it now takes into account whether or not a strike or spare occured in a frame. If so, it should correctly update the total score of the frame to reflect bonus points. Also, if it is the last frame – there are 10 frames in a bowling game - (or second to last frame) correctly allows 1 - 2 more "rolls" in order to correctly calculate the bonus for that frame.

In the **main** method, you need to:

- modify it so that it will accept 1 command-line argument that represents the number of players.
    - It should print out an informative message if you try to call it without passing an argument for the number of players.
- Create an array of Bowling_Frame pointers. Array should have 1 element for each player.
    - For example, a 3-player game would have a 1x3 array of Bowling_Frame pointers.
- use a loop to create 10 frames for each player, storing the first frame in your array
    - example:
      3 player array:
        1st frame for player 1 is at array[0]
        1st frame for player 2 is at array[1]
        etc…
    - Be sure to set links for all frames that belong together so that you don't lose the frames.
    - use "new" to create frames on heap
- use the new **update_scores** to set some scores on each frame
- print total game values using your helper function for every player.

Rubric

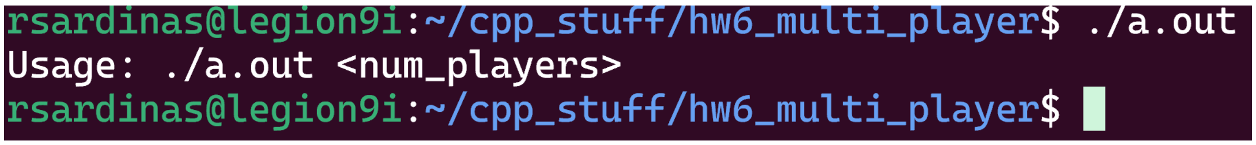
1. (10 pts) **update_scores** method correctly updates the total frame score, adding the bonus from other frames if a spare or a strike it thrown during the frame.
2. (10 pts) **update_scores** will correctly update the total frame score if a strike or spare is thrown and it is the last or next to last frame.
3. (10 pts) The **main** method accepts and uses 1 command-line argument to create and appropriately sized array for bowling players.
4. (10 pts) Your program will terminate and print an error message if you call it without passing in an argument on the command line.
5. (10 pts) Your program creates an appropriately sized array of pointers for bowling frames.
6. (10 pts) Your program creates 10 frames for each player in the array, storing the first frame for each player in the array.
7. (10 pts) Previous and Next pointers set correctly for all frames for each player (each player is a doubly linked list of bowling frames).
8. (10 pts) All frames created on the heap using the **new** keyword.
9. (10 pts) **update_scores** called an appropriate number of times to populate all scores for every frame in the game for all players.
10. (10 pts) Total game values printed for every player. Score for every frame is shown as well as a correct total.

Deliverables

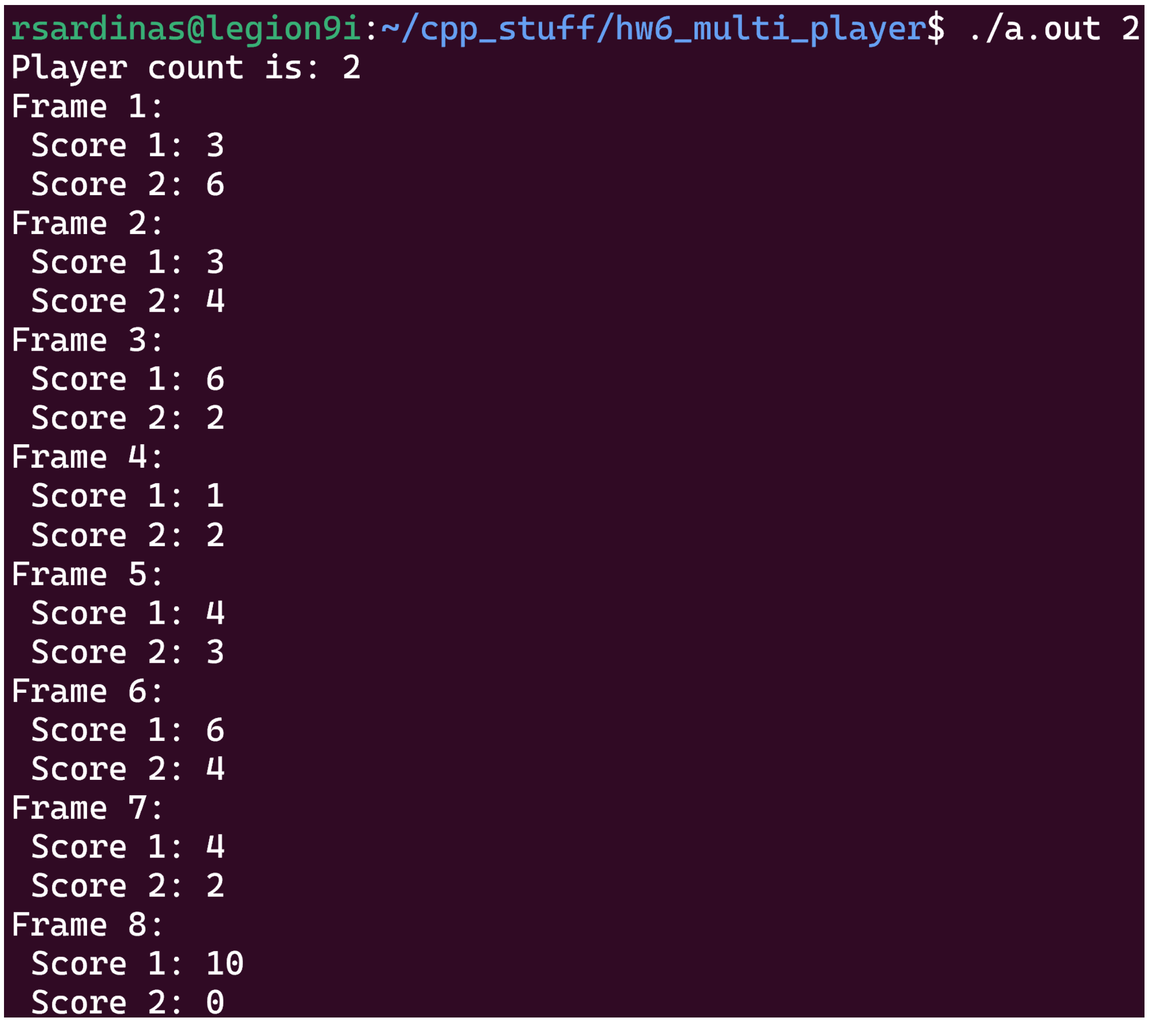
Your program MUST produce output. When you run your program in Jenkins, you should run it 3 times. Your output should look like the example below.

- Call it once and do not pass it output. (I need to see the error message).
- Call it once and pass it 2 players.
- Call it once and pass it 3 players.

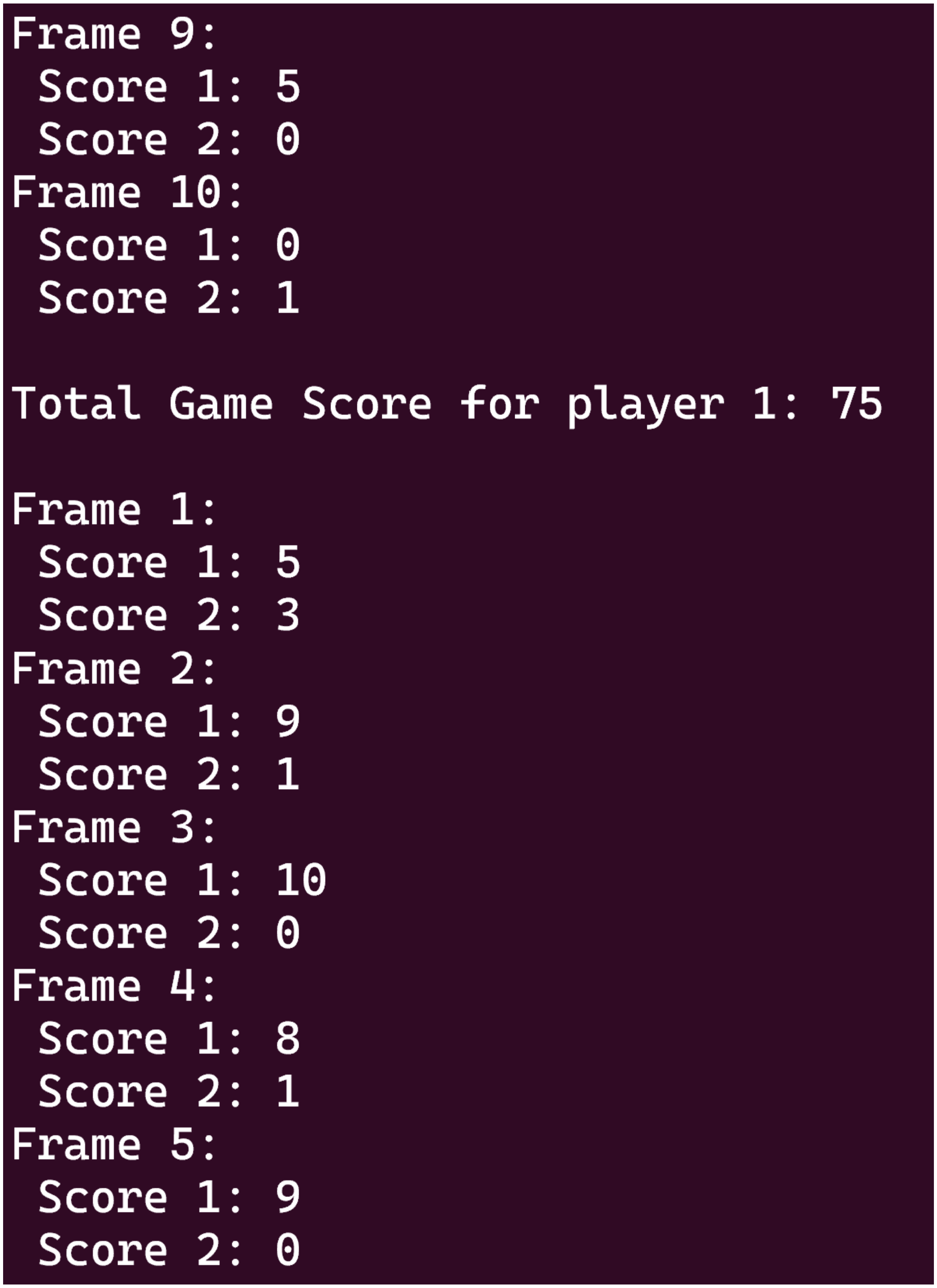
Sample Output

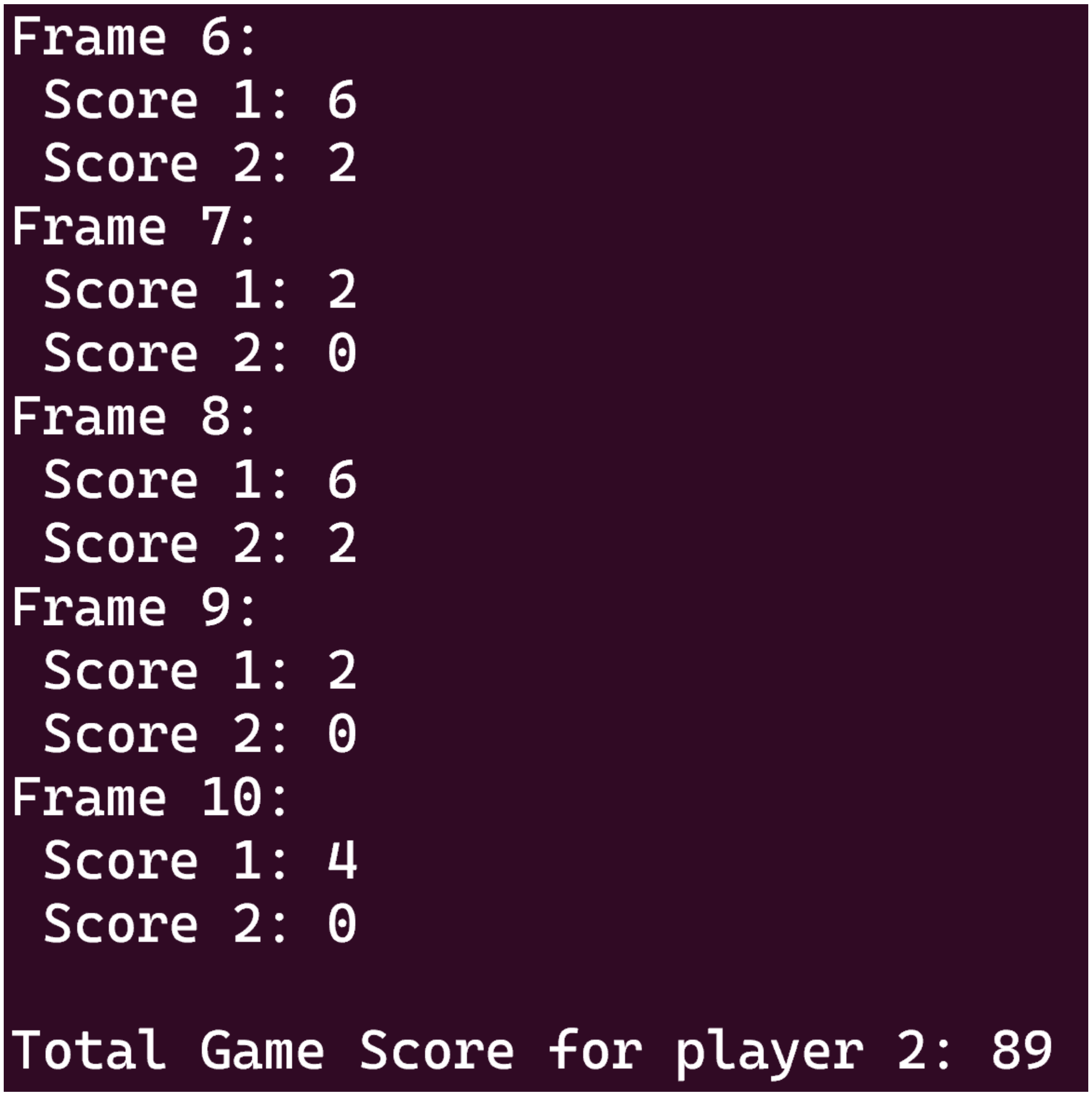


```
rsardinas@legion9i:~/cpp_stuff/hw6_multi_player$ ./a.out
Usage: ./a.out <num_players>
rsardinas@legion9i:~/cpp_stuff/hw6_multi_player$ 
```

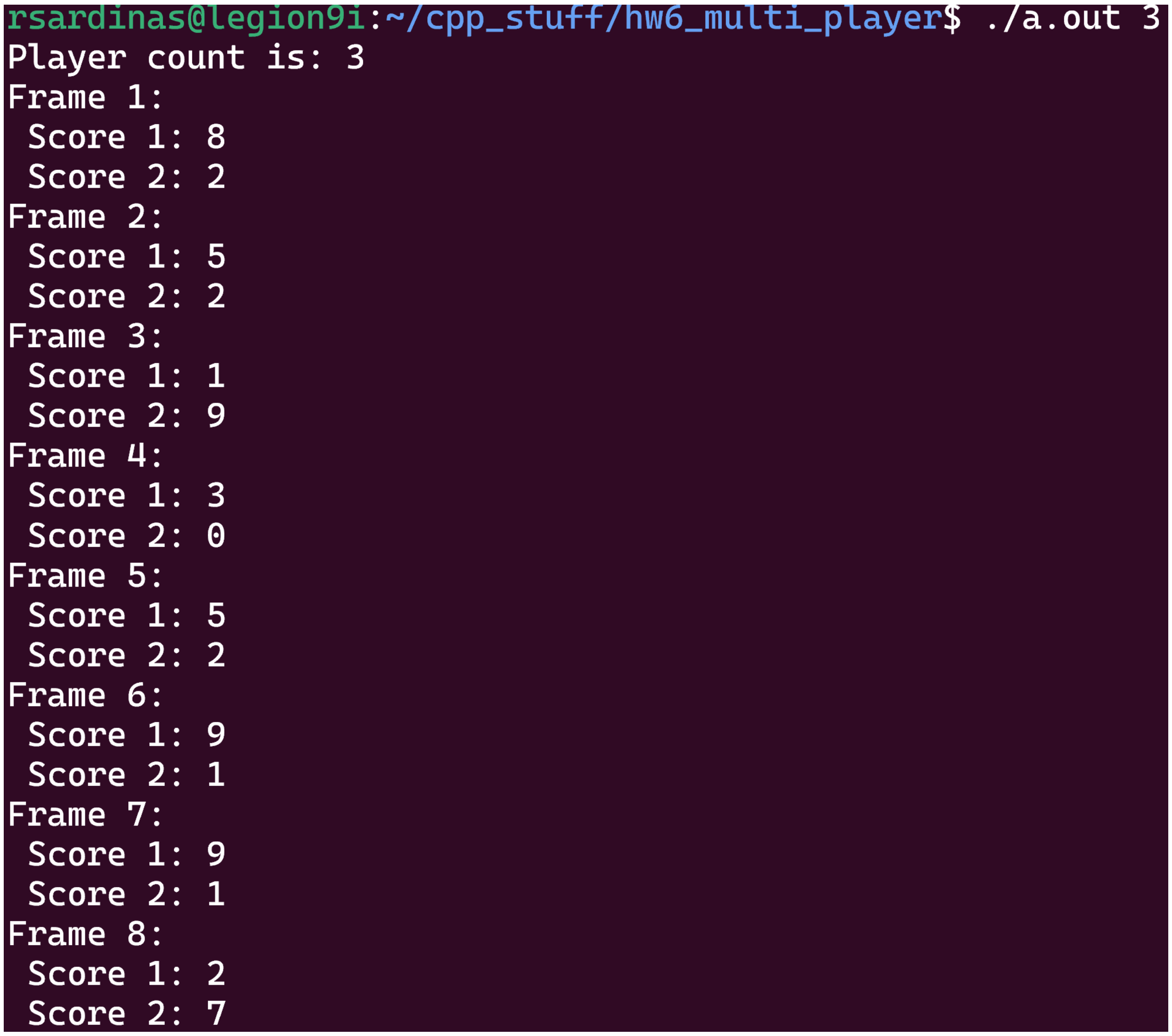


```
rsardinas@legion9i:~/cpp_stuff/hw6_multi_player$ ./a.out 2
Player count is: 2
Frame 1:
 Score 1: 3
 Score 2: 6
Frame 2:
 Score 1: 3
 Score 2: 4
Frame 3:
 Score 1: 6
 Score 2: 2
Frame 4:
 Score 1: 1
 Score 2: 2
Frame 5:
 Score 1: 4
 Score 2: 3
Frame 6:
 Score 1: 6
 Score 2: 4
Frame 7:
 Score 1: 4
 Score 2: 2
Frame 8:
 Score 1: 10
 Score 2: 0
```

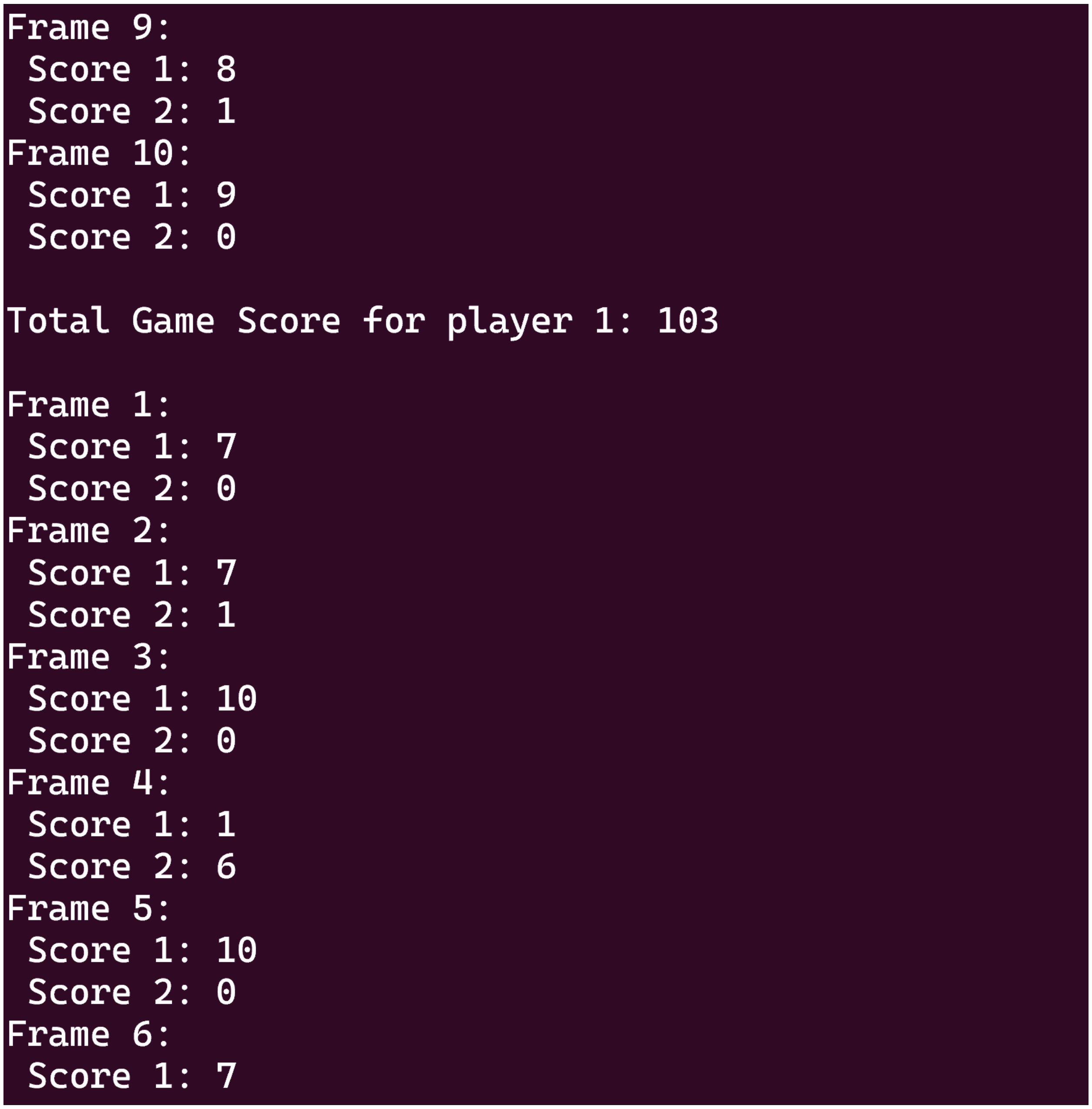

```
Frame 9:
 Score 1: 5
 Score 2: 0
Frame 10:
 Score 1: 0
 Score 2: 1

Total Game Score for player 1: 75

Frame 1:
 Score 1: 5
 Score 2: 3
Frame 2:
 Score 1: 9
 Score 2: 1
Frame 3:
 Score 1: 10
 Score 2: 0
Frame 4:
 Score 1: 8
 Score 2: 1
Frame 5:
 Score 1: 9
 Score 2: 0
```



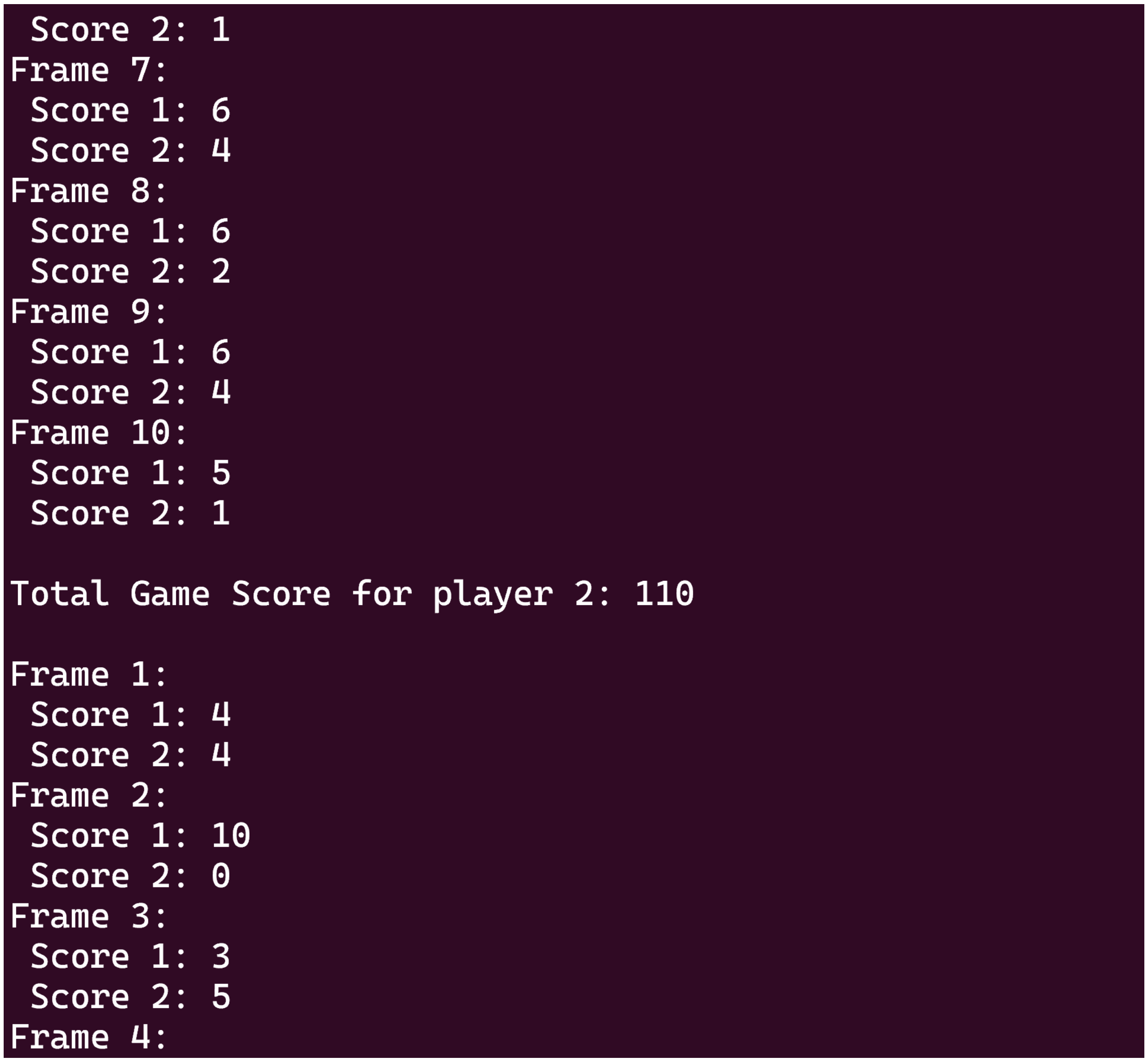
```
Frame 6:
 Score 1: 6
 Score 2: 2
Frame 7:
 Score 1: 2
 Score 2: 0
Frame 8:
 Score 1: 6
 Score 2: 2
Frame 9:
 Score 1: 2
 Score 2: 0
Frame 10:
 Score 1: 4
 Score 2: 0

Total Game Score for player 2: 89
```

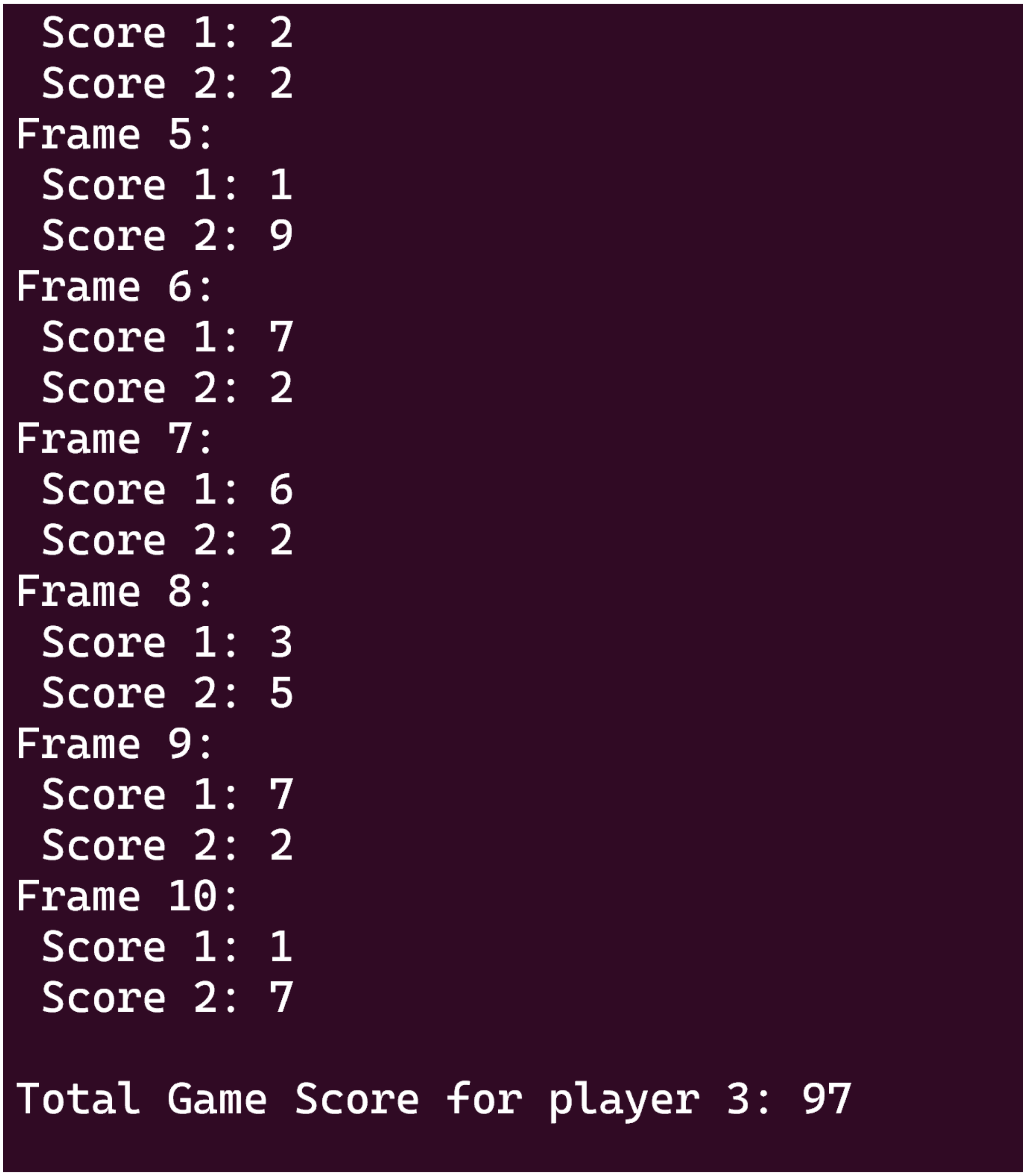


```
rsardinas@legion9i:~/cpp_stuff/hw6_multi_player$ ./a.out 3
Player count is: 3
Frame 1:
 Score 1: 8
 Score 2: 2
Frame 2:
 Score 1: 5
 Score 2: 2
Frame 3:
 Score 1: 1
 Score 2: 9
Frame 4:
 Score 1: 3
 Score 2: 0
Frame 5:
 Score 1: 5
 Score 2: 2
Frame 6:
 Score 1: 9
 Score 2: 1
Frame 7:
 Score 1: 9
 Score 2: 1
Frame 8:
 Score 1: 2
 Score 2: 7
```



```
Frame 9:
 Score 1: 8
 Score 2: 1
Frame 10:
 Score 1: 9
 Score 2: 0

Total Game Score for player 1: 103

Frame 1:
 Score 1: 7
 Score 2: 0
Frame 2:
 Score 1: 7
 Score 2: 1
Frame 3:
 Score 1: 10
 Score 2: 0
Frame 4:
 Score 1: 1
 Score 2: 6
Frame 5:
 Score 1: 10
 Score 2: 0
Frame 6:
 Score 1: 7
```


```
 Score 2: 1
Frame 7:
 Score 1: 6
 Score 2: 4
Frame 8:
 Score 1: 6
 Score 2: 2
Frame 9:
 Score 1: 6
 Score 2: 4
Frame 10:
 Score 1: 5
 Score 2: 1

Total Game Score for player 2: 110

Frame 1:
 Score 1: 4
 Score 2: 4
Frame 2:
 Score 1: 10
 Score 2: 0
Frame 3:
 Score 1: 3
 Score 2: 5
Frame 4:
```



```
 Score 1: 2
 Score 2: 2
Frame 5:
 Score 1: 1
 Score 2: 9
Frame 6:
 Score 1: 7
 Score 2: 2
Frame 7:
 Score 1: 6
 Score 2: 2
Frame 8:
 Score 1: 3
 Score 2: 5
Frame 9:
 Score 1: 7
 Score 2: 2
Frame 10:
 Score 1: 1
 Score 2: 7

Total Game Score for player 3: 97
```