PROJECT TITLE: SNAKES AND LADDER GAME

Team Members:

Sl. No	Name	Roll No.
1	Samriddha Chattopadhyay	200020040
2	Naram Keshav	200020029
3	Samir Bhauraoji Giripunje	200030049
4	Bondugula Pranav	200020010

Problem Statement:

Write a C program to implement the snake and ladder game described below.

a) Consider a grid of 10x10.

b) The starting point is number '1' and winning point is number '100'.

c) The players must throw a dice (in this case enter a number between 1-6). Based on the number entered the respective player's pawn is either moved forward or backward based on whether a ladder is present or a snake is present.

d) A file snakes.txt should hold information on the number of snakes present, length of snake and position of snake. For example - 4 small snakes with length 10 units and 2 large snakes with length 15 units can be considered. The length indicates if the snake head is at number 26 then tail would be at 16 for a small length snake.

e) A file ladder.txt should hold information on the number of ladders present, length of ladder and position of ladder. For example - 3 small ladders are present in the numbers 5, 33, 89. Small ladders are of the length 10 units. i.e if the base of the ladder is at 5 then its tip will be at 15. 2 large ladders are present in numbers 29 and 74. Large ladders are of the length 20 units. I.e if the base of the ladder is at 29 then its tip will be at 49.

f) When the player reaches number 100, the winner must be declared.

Design Steps:

1. The main code has definitions and declarations of functions. Each function implements the features mentioned in the problem statement. A welcome screen is displayed at the beginning of the game, where the number of players and player names is taken as input.
2. Global variables are used to store the length, position and number of different types of snakes and ladders (i.e., one three variables each for small snakes, long snakes,small ladders and long ladders)
3. A structure for players to store their names and position as input (positions are initialised to 0).
4. File_Input function to take input from the text files and provide the appropriate data for snakes and ladders from the text files.
5. Usual_Process function to handle the usual game methods, i.e., moving forward when a number is inputted or a ladder is encountered, and moving back when a snake is encountered.
6. Corner_Cases for handling corner cases like bounceback (explained below) when a player seems to move more than 100 or when more than one six comes up or when three sixes come up, etc.
7. A custom header file to fix the positions of snakes and ladders (i.e., to implement the dynamic snakes and ladders feature described below).
8. Some additional header files to add some delay and to enable the **press any key to end the game** feature (described below).

Additional Features:

- ➔ **Checking Input Validity:** Checking if input is within 1 to 6 (both inclusive) in each input.
- ➔ **Opening with a one**: A player has to come up with a 1 in order to start the game.
- ➔ **A twist in the last feature:** When a 1 comes up for a player, that player is allowed to move, but he doesn't move in that turn. He can progress forward from the next turn
- ➔ **Multiple sixes rule:** If a six comes up, the player gets another chance to roll the dice after moving six places. If another six comes up, the player can roll a dice again after moving six more places. However, if a six comes up a third time, the last two sixes along with this six will be cancelled, and the player will be sent back to the initial position and asked for a dice input again (initial position means before throwing 3 continuous sixes). For example, if a player is at 30, and he gets a six, he will move to 36 and will be asked for an input. Now if he gets a six, he will move to 42 and he will be asked for an input again. However, if he gets a six again (third six), then he will be sent back to 30 and asked for an input again.
- ➔ **Bounceback:** If a player is in a position, where his progress is limited, like in the case when player is at 96 (and he needs an exact 4 to win), if the player gets an input which will make him cross 100 by some places, then, the player moves to 100, then goes back by those many places. For example, if a player is at 98 and he gets a five, as 98+5=103, he will go to 100 first, then come back to 97, as, 103-100=3 and 100-3=97
- ➔ **Dynamic Snakes and Ladders:** The position of the snakes and ladders are changed after every game, and the positions are decided randomly. So noting down the positions of snakes and ladders and avoiding or falling on them on purpose is prohibited.
- ➔ **Scoreboard:** When the game starts for the first time, Rues and welcome screen will be shown, however, as soon one of the players gives an input, a scoreboard will be shown noting down the positions of each player along with the name. This helps in tracking the position along with the messages popping up in the terminal.
- ➔ **Precise Display of Information:** Unlike earlier, the game will now show the details of only the current player, and there will be a 5 seconds gap before the turn of each player. This is primarily implemented to make the display

of the game clean with fewer but precise information, rather than showing the information of all players and letting them scroll up to see their previous positions.

➔ **Press any key to end the game:** On pressing any key after the game ends will clear the terminal of any data that had been shown during the game. This gives a feel of closing a game that has a UI (although this lacks a UI).

Role of team members:

Student Name	Role
Samriddha Chattopadhyay	Main function, Making of additional header files and Addition of padding in some places
Naram Keshav	Welcome function and File_Input function
Samir Bhauraoji Giripunje	Usual_Process function and search functions
Bondugula Pranav	Corner_Cases function, Bounceback implementation and Delays

References:

- For libraries https://cplusplus.com/reference/clibrary/
- For implementation of random function https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/
- For brushing up file handling concepts https://moodle.iitdh.ac.in/pluginfile.php/13836/mod_resource/content/1/09_files.pdf
- For addition of appropriate padding in some places https://drive.google.com/file/d/17ibGia3CUvyLoDYdaiQU3wcNW4RvPwT0/view?usp=sharing

Results:

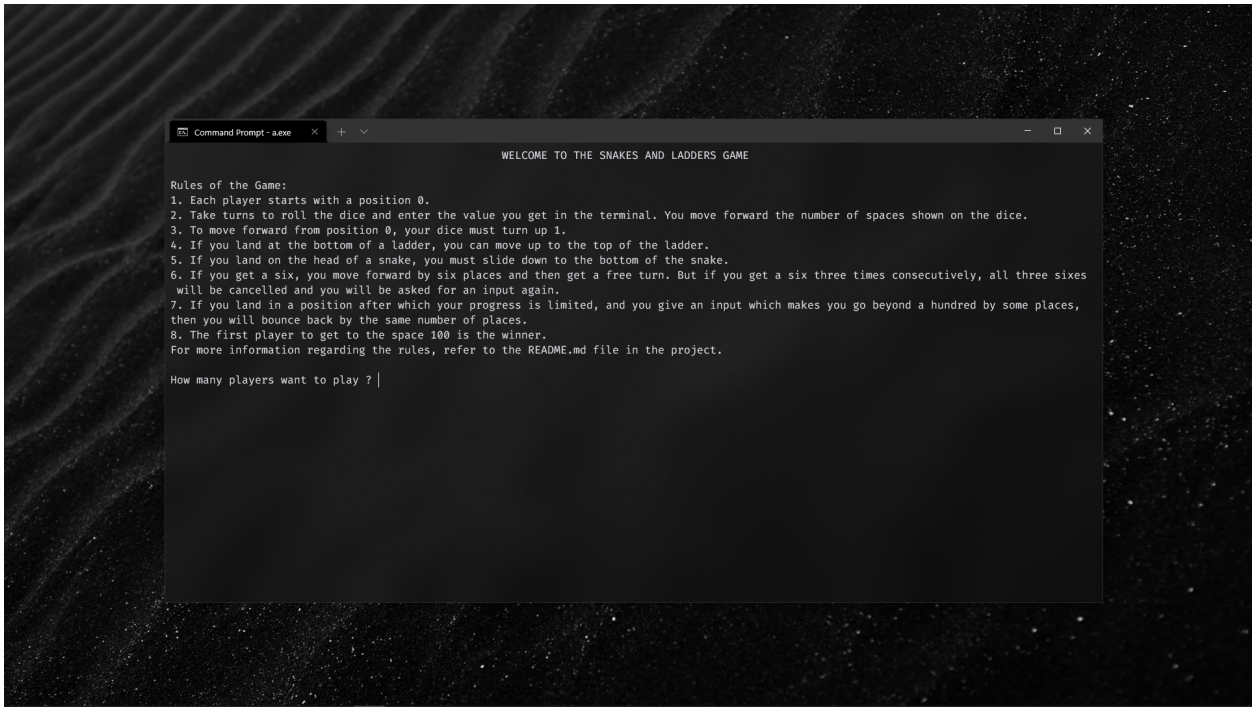

Fig 1 Depicts the welcome screen for players before starting the game

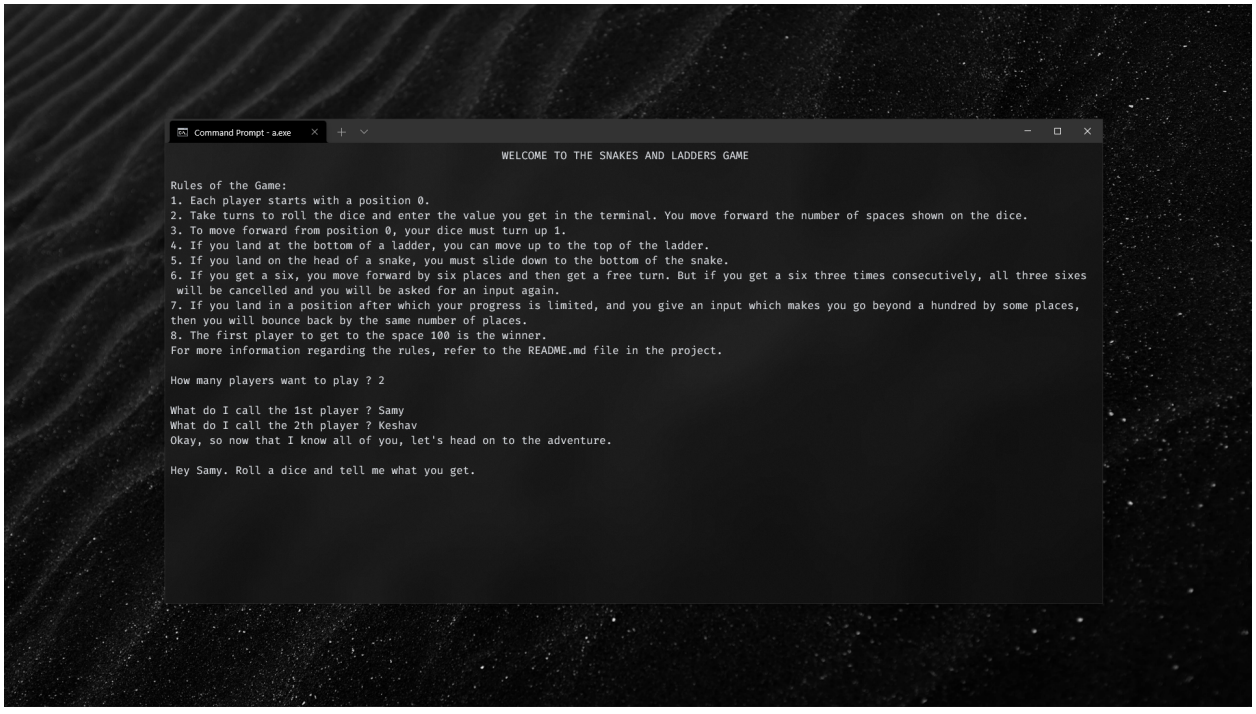

Fig 2 Depicts the welcome screen for players before starting the game

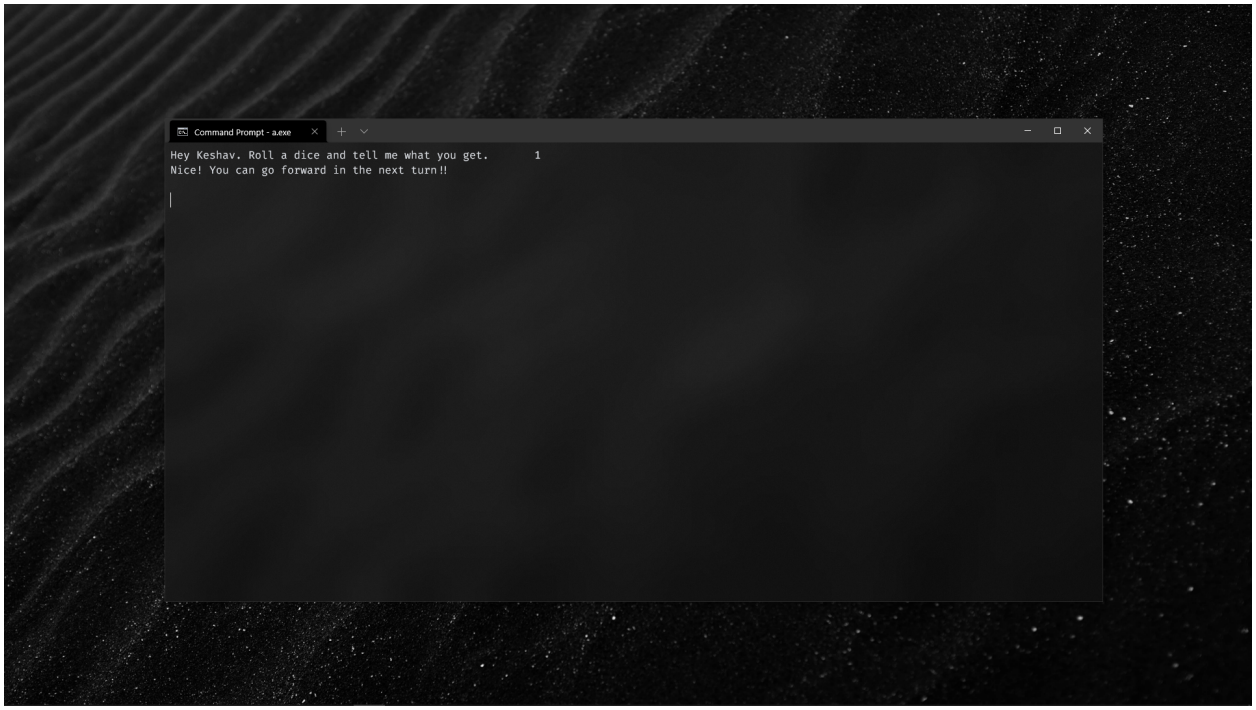

Fig 3 Depicts the opening with a one rule

Fig 4 Depicts the rejection of invalid input

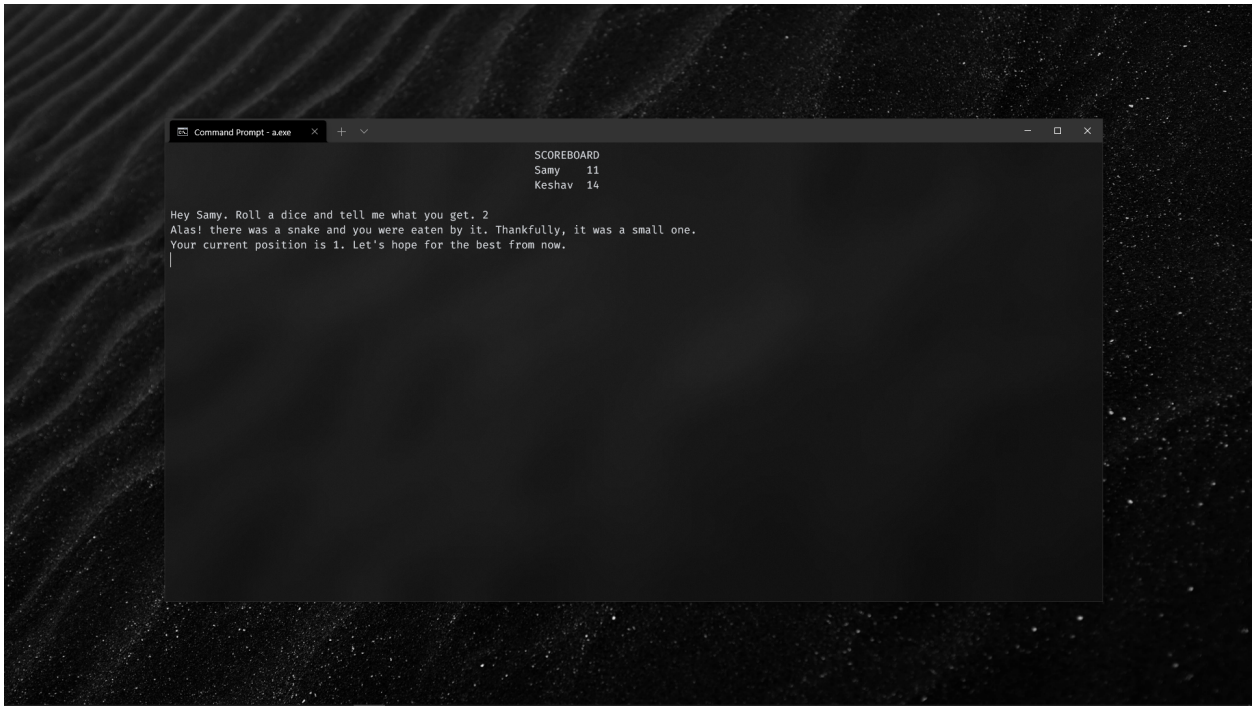

Fig 5 Usual rule of the game where the player goes back if he reaches a snake

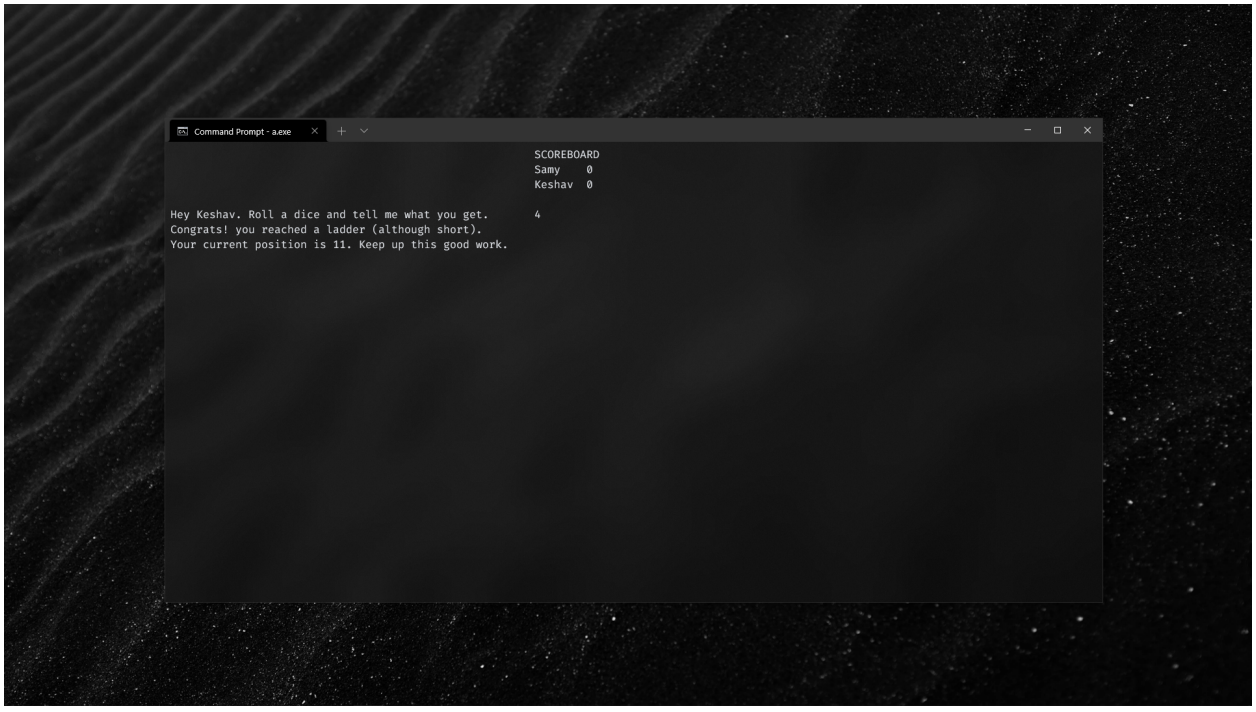

Fig 6 Usual rule of the game where the player goes ahead if he reaches a ladder

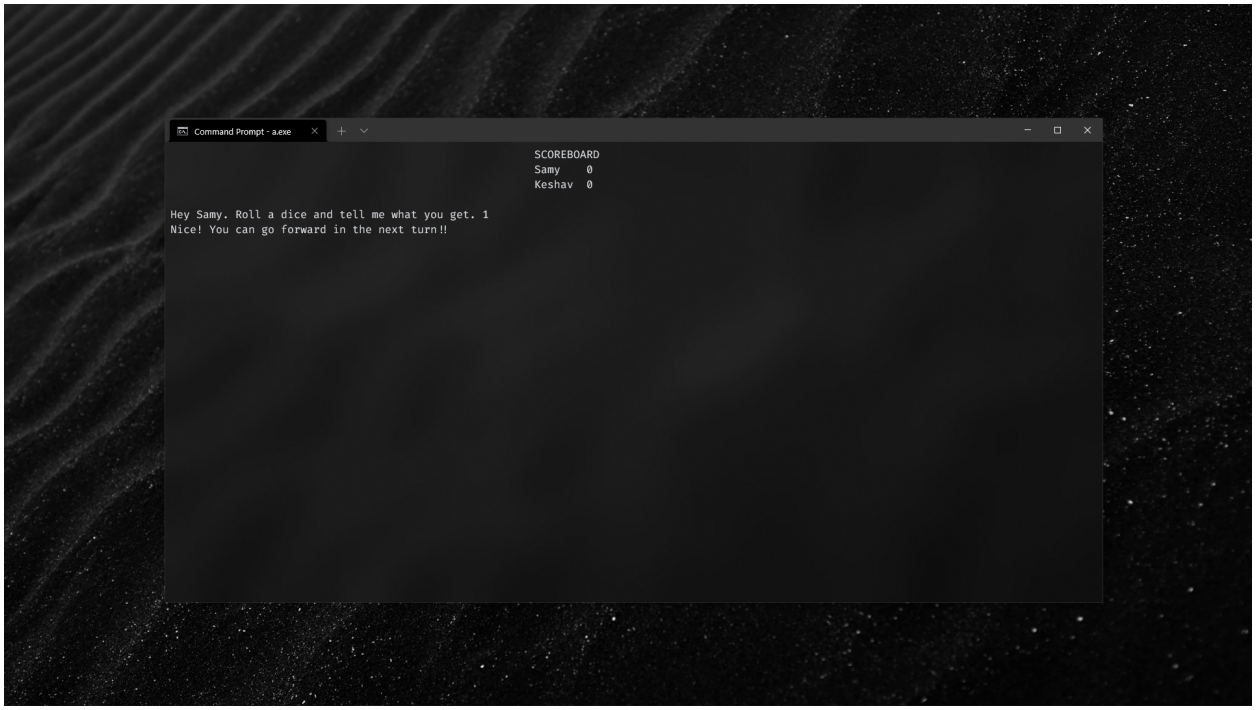

Fig 7 Depicts the scoreboard

Fig 8 Depicts what happens when player gets one six

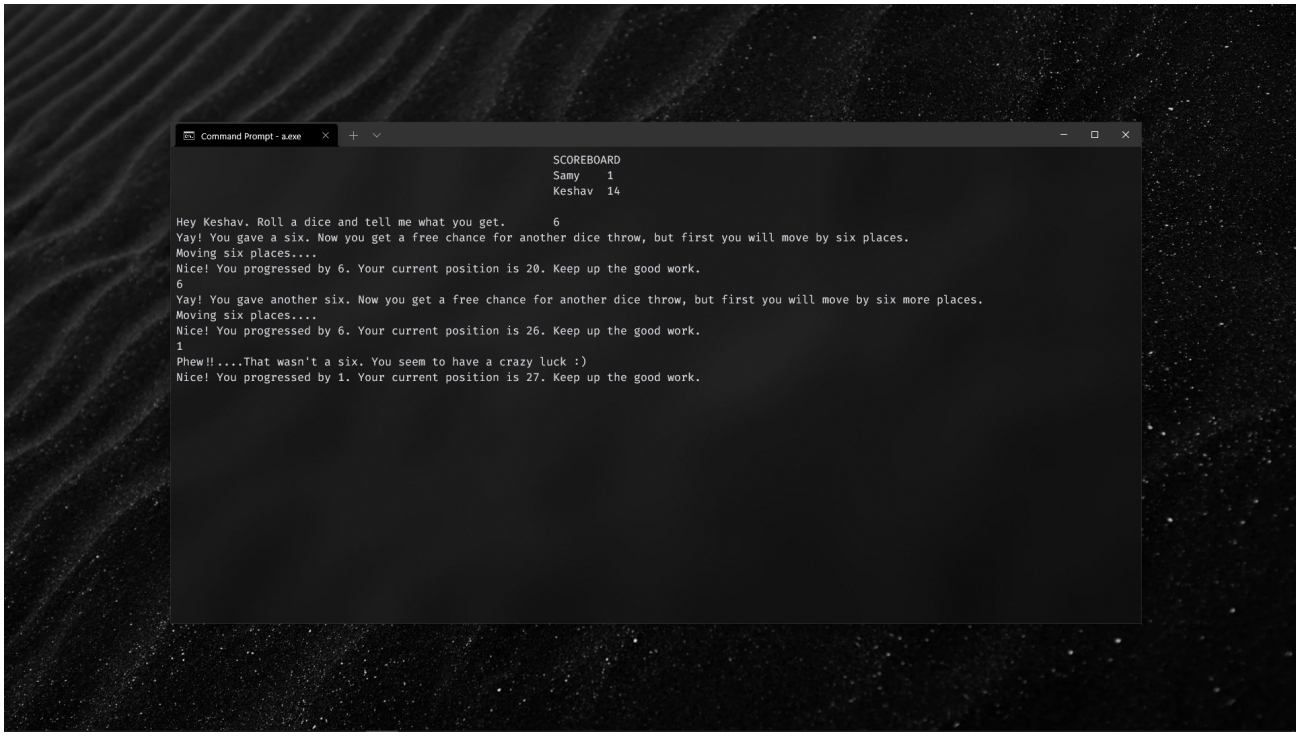

Fig 9 Depicts what happens when player gets two sixes

Fig 10 Depicts what happens when player gets three sixes

Fig 11 Depicts the bounceback rule

Fig 12 Depicts declaration of the winner as soon as someone reaches 100