Algorithm for the Position Function

1. Call position snake and position ladder, where each function does the following.
2. Generate random numbers between a specified upper bound and lower bound.
3. The numbers generated are assigned to various variables and array elements, depicting length, number and position of snakes and ladders.
4. Finally, respective files are opened and the values are written in it, which are to be read by the file input function in the main function.

Algorithm for the Welcome Function

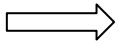
1. Clear the screen.
2. Print welcome message and rules.
3. Ask for number of players.
4. Run a for loop and take input for each player in an array of structures containing player name and player position.
5. End the function by printing a starting message

Algorithm for the File Input Function

1. Open the Snakes.txt file.
2. Store the lengths of the snakes from the file in respective variables.
3. Store the number of the snakes from the file in respective variables.
4. Store the positions of the snakes from the file in respective arrays.
5. Do the same for Ladders.txt.

Algorithm for the Usual Process

int position, int option



If position + option is less than 100.

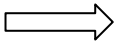
    Add option to position.

    If snake at position + option, go down and if ladder at position + option, go up

Else go back by position + option - 100

Algorithm for the Corner Cases

int position, int option



If one six comes up, Store the initial position, call usual process with 6 and ask for another input.

    If another six comes up, call usual process with 6 again and ask for input.

        If another six comes up, restore the position to initial position and take input again. If this is a six, call the corner cases function recursively, else call usual process with current input

    Else, call usual cases using current input.

Else, call usual cases using current input.

Algorithm for the Main Code

1. Call the position function to position the snakes and ladders
2. Call the Welcome function to take player inputs and show the welcome screen
3. Call the File Input function to take data from files containing snakes and ladders information
4. Run a while loop, with condition as position of ith player will not be equal to 100.
    a. Take dice input
    b. If input is invalid, show a error message and ask for input again.
    c. For valid input, check if player is at position 0. If that's the case, then check if input is 1. If it is 1, let make player position -1 otherwise, go to the next player.
    d. Check if player position is -1. If it is -1, make it 0 and Call the usual process function or the Corner cases function based on the whether input is 6 or not.
    e. In all of other cases, check if user input is 6. If it is 6, call the corner cases function. Otherwise call the usual process function
    f. Go to the next player.
    g. For each Iteration, clear the screen and update and print the scoreboard.
5. When some player reaches 100, the loop breaks and the winner is declared.
    a. To declare the winner, search for the player with position 100 in player array and return the player after it is found.
    b. Store the data in a winner variable, and display it in the terminal.
6. Now, display some ending message and call the getch() function from the conio.h library to end the program at keypress.