Snake in C/C++

As a player, I want to play a classic snake game on my computer, so that I can enjoy a fun and challenging gaming experience.

Acceptance Criteria:

1. The game should have a grid-based layout where the snake moves within boundaries.

2. The snake should start with a single segment and grow longer as it eats fruits.

3. There should be a fruit randomly placed on the grid for the snake to eat.

4. The snake should move in four directions: up, down, left, and right.

5. If the snake collides with the boundary or itself, the game should end.

6. The score should be displayed on the screen, increasing as the snake eats fruits.

7. The game should provide an option to restart or quit after game over.

8. The game should have simple graphics and smooth movement for a good user experience.

Tasks:

1. Set up the game environment with a grid and initialise variables for the snake, fruit, score, and game state.

2. Implement logic for snake movement, collision detection, and fruit consumption.

3. Design user interface elements such as score display, game over screen, and restart/quit options.

4. Test the game thoroughly to ensure smooth gameplay and correct functionality.

5. Optimise the code and improve any areas for better performance and user experience.

6. Deploy the game for users to play and gather feedback for further enhancements.

Detailed Work Breakdown:

1. **Setup Function (setup):**
   - Initialises the game variables such as snakeposition(x, y), fruit position(fruitx, fruity), score, and gameover both initialized to 0.
   - Randomly places the fruit (*) within the game boundary. (use rand function)
2. **Draw Function (draw):**
   - Clears the console screen (system("cls")) and draws the game grid.
   - Draw drawBoundary(length, breadth,'*'), drawSnake(x, y,'@','o') and drawSnakefruit(fruitx, fruity,'#') (Uses # for boundaries, 0 for the snake's head, and * for the fruit).
   - Prints the current score and instructions to quit the game.
3. **Input Function (input):**
   - Checks for keyboard input using kbhit() and reads a character using getch().
   - Maps keyboard inputs to snake movement: 'a' (left), 's' (down), 'd' (right), 'w' (up), 'x' (exit).
4. **Logic Function (run):**
   - Moves the snake according to the flag (direction) taken from the **input function**.
   - Handles boundary conditions to determine if the snake hits the wall and ends the game (gameover = 1).
   - Checks if the snake eats the fruit (*) and updates the score accordingly.
5. **Main Function (main):**
   - Initialises the game setup.
   - Enters a game loop that continues until gameover becomes true.
   - In each iteration of the loop, it calls draw to display the game through draw, input to get user input, and logic to update the game state through run function.

```
int main(){

        setup();
```

```
		while(!gameover){
			system(“cls”);
			draw();
			input();
			run();
			usleep(400000);//to control speed of snake
		}
	}
```

Snake in C++ class abstraction:

```
class Snake{
        private:
                // Declare your Variables here

        public:
                //constructor for variables initializations
                Snake(){
                        setup(); //call setup to initial setup
                }

                //Define your other methods here
                setup(){
                        //write your code here for setup
                }

                draw(){
                        //write your code here for drawing screen
                }

                input(){
                        //write your code here for getting input from keyboard
                }

                run(){
                        /*write your code here for logic building for snake movement,
                        regeneration of food, increasing length of snake, increasing snake speed
                        and identifying the collision and stopping the game*/

                }
};

// define main function here
int main(){
        Snake s;
        while(!gameover){
                system("cls");
                s.draw();
                s.input();
                s.run();
                usleep(400000);//to control speed of snake
        }
}
```