Milestone 1:

Bunea Nicolae

The main goal of ***Azul*** is to score the most points by strategically placing tiles on your personal player board to complete patterns and lines.

How to setup the game:

- Each player will receive a player board and 1 scoring marker that would be placed on ‘0’, on the score track (top-left).
- We’ll place the Factory displays (the honeycombs) in a circle around the center of the table (5 ones for 2 players, 7 for 3 players and 9 for 4 players).
- From the bag, where there are 100 pieces (20 of each color: RED, BLACK, WHITE, BLUE, YELLOW), we extract and place exactly 4 tiles randomly on the Factory displays.

The game, which is played in multiple rounds, is composed in 3 phases:

1) **Factory offer**, consisting in picking tiles from the honeycomb:
    - You have to pick all tiles of the same color from any Factory display, and the remaining tiles will be moved to the center of the table.
    - Or, you could pick all tiles of the same color from the center of the table. If you are the first player from this round to pick from the center, you have to take the starting marker and place it on the floor line (left-bottom).

    - add the tiles you picked to one of the 5 pattern lines on your player board, one by one from right to left.
    - if the pattern line already holds tiles, you can only add the same color to it.
    - if all the spaces are filled, and you have more tiles that you can place on the chosen pattern line, you have to put the excess tiles on the floor line, being penalized at the end of the round for them.

2) **Wall-tiling**, consisting in moving their completed pattern lines to their walls:
    - from top to bottom move rightmost tile of each completed line in the corresponding line of your wall
    - then, remove all the tiles from any pattern line that was completed before and move them in the bag.
    - after that, any remaining tiles on the pattern line will stay on your player board on the next round.

    - the scoring is given immediately after placing on the space matching as following:

- if there are no tiles directly adjacent to the placed tile, you gain 1 point (on the score track)
- if there are any tiles adjacent to the placed tile, count all the linked tiles (including the newly placed one) and receive points for every linked tile (EX: 3 points for 3 horizontally linked tiles, 2 points for 2 vertically linked tiles, or these could combined themselves if they're linked both horizontally and vertically).

- at the end of the wall-tiling phase, check if you have any tiles on your floor line, and for each tile you have there, you lose the number of points indicated above it.

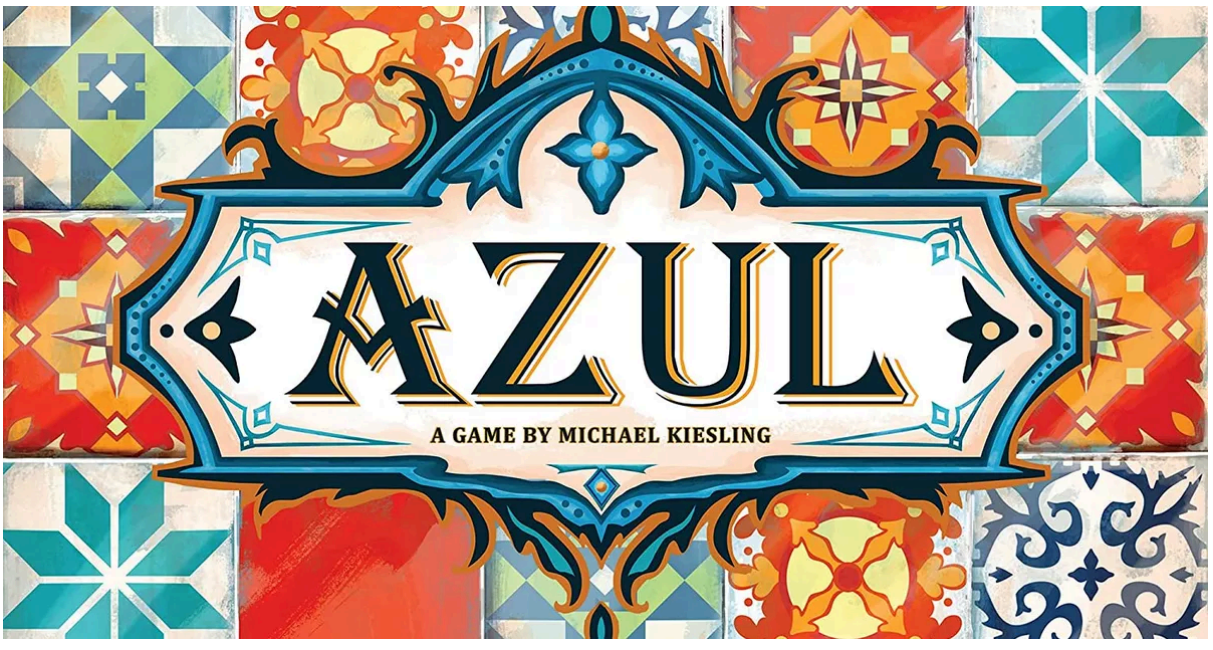
- all these tiles will be placed again on the bag after.

3) **Preparing the next round**:
   - If nobody completed **a horizontal line of 5 consecutive tiles**, the player with the starting marker will refill each honeycomb and will start the next round.
   - Everyone will play the game as before, until someone will complete one of the horizontal lines and the **endgame** begins.

The endgame:

- The game ends right after someone has completed 1 horizontal line in the Wall-tilling phase and everyone will gain their additional points for the following:
  - 2 points for each completed **horizontal** line.
  - 7 points for each completed **vertical** line.
  - 10 points for each color of which you placed all 5 tiles on your wall.

The player who has the *most points on his score track* **wins the game**!



Technical Outline

In order to implement Azul in C, I have some functions in mind to make it, structured in manageable components by functions:

1) *Game Initialization:*

- ***bag_contents() - A function that stores the bag with all the tiles:***
  - *Each tile could be denoted with a number to be stored: 1 – red, 2 – blue, 3 – orange, 4 – black and 5 – blue with white. (Could be an array)*
- **game_initializer() - this function would handle the setup of the game:**
  - including initializing variables, setting up the game board, distributing tiles to the factory displays, and assigning player boards to each player.

2) *Player actions + Input commands:*

- **players_counter() - A simple function to know how many players we have:**
  - select at the start of the game the number of players: 2, 3 or 4
- **board_display() - this function would display the current state of the board:**
  - including the factory displays, player boards, and scores, allowing players to see the game state at any point during the game.
- **tile_offer() - this function would manage the factory-offer phase of the game:**
  - It could handle player turns, allowing them to choose tiles from factory displays or the center pile, and updating the game state accordingly.
- **tile_placement() - this function would handle the tile placement phase of the game:**
  - It would allow players to place tiles onto their player boards, checking for valid placements and updating scores and board states accordingly.

3) *Game end + output:*

- **end_round() - this function would manage the end of each round:**
  - including discarding excess tiles to the floor line, updating the starting player, and checking for the end game condition
- **score_counter() - this function would calculate the final scores:**
  - at the end of the game, it would take into account completed rows and columns, bonuses, and penalties.
- **endgame_check() - this function would check for the end game condition:**
  - determining if any player has completed a horizontal row on their player board, and triggering the end of the game if necessary.

Relevant data structures

❖ A player structure:
  - to store information about each player, such as their name, current score. player's pattern line, floor line and wall.

- A game board structure:
    - to store information about the factory displays, how many and what tiles will be shown on them.
    - maybe the bag of tiles, the number of tiles in it and an array of players that plays the game.
- A player board structure:
    - to store information about the player's tiles placed on it, including color and position.
    - the row of tiles placed on the pattern line
- Some settings for the game in a structure:
    - to store the game settings as the number of the players, the number of rounds and maybe any other configurable options that could be additional in the future.