Algorithm for the game

Step 1: Create a method **characterCharacteristics ()** with 4 Characteristics (STRONG, STABILITY, DETERMINED and INTUTIVE) and each characteristics with random values.

Step 2: Create a method **Split_cards ()** for creation of deck of 10 cards, each card representing a character and further on shuffling all the 10 cards and distributing it within 2 players.

Step 3: create a method **dice ()** for deciding which player should start the game. The player with highest number on the card wins the chance to play. The player with highest number on the dice is winner.

Step 4: Create two methods **Player1 ()** and **Player2 ()** for starting the game on the basis of the step 3.if player1 wins the step 3 call the function player1 () and if player 2 wins the step3 the call the function player2 ().Each function will give input for the start of the game, select the character , display the characteristics and challenges the next player with one characteristic from any of the four characteristics.

Step5: Create a method **comparison ()** which will print for the challenged player all the four characteristics and evaluate the value for the challenged characteristics.

Step 6: Create a method **evaluator ()** for evaluating which player has highest value for the chosen characteristic and will display the winner for that round.

Step 7: create a method **Strategy ()** for staging method Player1 () and player2 () on the basis of the winner of step 3.

Step 8: Create a method **outdated deck ()** for appending all the played cards/Characters in the first round.

Step 9: create a method **stratergy2 ()** for staging Player2 () and Player 1() on the basis of the winner in step 6. A while loop condition to be created for running all the cards left in the deck.

Step 10: Create a method **final Count ()** which will give count of all the players won in each round and displaying final winner of the game.

Step 11: Create a method **resurrect ()** which allows user to play with outdated cards with the help of outdated deck () and play only after step 6. This function will accept the card no to be played on the outdated deck and also the characteristic. This method can be used only once with each player.

Step 12: create a method **resurrect strategy ()** for staging of the player on the basis of the winner in Step 6. **The resurrect ()** spell chance for the player will be decided on the basis of the winner from the step 7.

Step 11: Create a method **godspell1 ()** which allows user to play with random card to be chosen from the **deck ()** and play only after step 3. This function will also accept the characteristic to be chosen in advanced and challenged the opponent player to play with the chosen random card and also the characteristic. This method can be used only once with each player.

Step 12: create a method **godspell _strategy ()** for staging of the player on the basis of the winner in Step 3. The **godSpell_strategy 1()** chance for the player will be decided on the basis of the winner from the step 3.

Step 13: Create a method **game()** for staging all the methods in and declares the the final winner of the game.

godspell1 (): Helping functions used **godspell2 ()** and **godspell _strategy2 ()**

Outdated deck (): Helping functions used **outdatedround1()** and **final_outdated()**

final_count(): helping functions used Starategy1_evaluator() and