Detail.

Project is incomplete. This a partial submission. Due to unforeseen circumstances out of my control that hindered me finishing in time. There were bad electricity blackouts from load shading over the weekend, as a result I did not get enough time to work on my assessment as planned. My sincere apology. But I will continue to with the assessment overnight and hope and hopefully be done by the morning. However, the partial design and game rules are explained before.

1. Game design
   1. Game rules (inspired from Russian roulette)
      1. Pick a card from screen, user can pick more than 1
      2. Each card has a probability of 1/36 for a win
      3. Bet per number is always one
      4. Press play to shuffle cards
      5. Win is awarded is the card from the shuffle matches a selected card.
      6. Button layout is as below.
      7. 
   2. To be implemented
      1. Win animation, current win animation plays a spiral around the winning card, the win meter is updated soon after the card shuffle finishes. A pop up on screen must be implemented to display the win in big digits in the middle of the cards grid.
2. Question 1.
   1. Easing the control of the motion of animations by affecting the rate at which animations animations are played ( acceleration, deceleration).
   2. Types of Easing:
      1. Ease -in
      2. Ease-out
      3. Ease - in and out
      4. Linear
   3. Possible use: On fading animations in and out / hiding and showing, to give a more realistic effect than a simple linear appearance and disappearance.
3. Question 2
   1. A promise is an object that may produce a value in the future, (an outcome from an attempted execution, like network sockets connectivity checking ).
   2. Advantages
      1. Can be used to mimic asynchronous code
      2. Provides better error handling as well as success handling
   3. Disadvantages
      1. Once a promised is settled, the outcome cannot be checked but rather an exception is thrown than to do a continuous check if it was required
4. Not sure about the question, but,
   1. Maybe using CPU cache for images that are on screen for long so that they are not redrawn all the time. The point will be to reduce draw calls.
   2. Frustrum culling, hiddin images that are not in the view of the camera. So in Phaser we can set the visibilty to false and the alpha to zero of images that are not in camera view.
   3. Sometimes one can reduce the resolution of images for faster computation, also save on memory. Even if the images lose their actual size, they can later be scales up, but on rendering, they are rendered faster. There will be lost in picture quality (disadvantage).
   4. Flat images can be rendered faster
5. Question 4:
   1. Creational
      1. This is specifically for design of classes that can be accessed from different parts of the software. Therefore the class can be instantiated from different parts part of the code and then extended to override methods etc
   2. Structural
      1. This method is best when creating large software structure that requires providing compatibility between objects and classes. Eg , interfaces, interfaces that are incompatible, this design pattern can be used to establish a relationship.
   3. Behavioral
      1. This is design based on the inter-dependency of classes and objects. This pattern can be used to better design the flow of software based on which classes or objects may have to control other classes and object. Eg instantiating other classes inside other classes for better software development.
6. This is a design pattern when and logic is defined in one function and the function will take in an object which defines a state. The function will return a new state (the transitioned to state based on the attributes of the input state).
7. Question 6
   1. This a loop that controls the overall flow of a game. This loop is continuously executed until the user terminates the game.
   2. FSP is = Frames per Second
      1. This is the rate at which frames are played. So the number of frames in a second (iterations per second, in Phaser it could be the rate at which the Update() function is called).

function convert(array)

{

    var split\_array = array.split();

    var result = [];

    for( i = 0; i < split\_array.length; i++)

    {

        result[i] = split\_array[i];

    }

    return result;

}

        private addArrayNumbers(array:number[]): number

        {

            var result:number = 0;

            for(var i:number  = 0; i < array.length; i++)

            {

                result += array[i];

            }

            return result;

        }