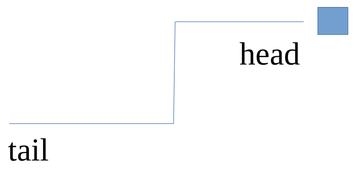
<u>1.Defects faced while the time of development</u>

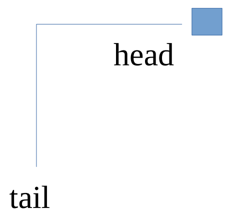
(I) **<u>The movement</u>**: At the beginning stage i was using some algebraic calculations for the snake movements. That is only fine for single cell movement when dealing with the multiple cell movement this is not an appropriate method.
The idea i was implemented is to use width and hight of the fillRect to do the calculations ie, when the snake moving upward direction the hight is incremented and width is decremented also in upward to rightward direction the snake hight is decremented and and width is increased.
So an alternatine method i used an array queue(FIFO) to store all the postions and at the time of movement use array popping to store all the new postions. hence the movement problem is solved.

(II) **<u>Left and right movement</u>** : Ie, if the snake is moving right direction the left movement needs to be disabled vice versa and if the snake is moving up direction the down needs to be disabled vice versa.

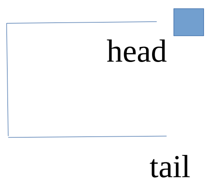
(III)**<u>snake eating food and snake length grows</u>**: At the initial stage when the snake eats food i tried to create a new cell in the tail of the snake but it is so hard to find the last element at that time. Ie, consider these are the three states of the snake



(i)



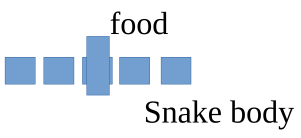
(ii)



(iii)

in these three cases the head is moving in the right direction but the tail is in defferent positions so to find the tail postion id really difficult. So as an alternative method i used to add the new element in the head and swapping all the elments to one postion down. But this also increases the time complexity of the program. So instead of adding a new element in the head and swapping all the elements i just tried to increase the length of snake when the snake eats the food.

(IV)**<u>Creating random food element</u>**:At the time of creating random food element i tried to use math function to generate random element but in this case at some stages the food was created at the same postion of the body of the snake. So before creating random element i checked if it is existing in the body if so create new random element which is not containing in the body. hence that problem is solved.



Experience Report

HTML

Hypertext markup language is used to design web pages and that can displayed in any web browsers. It can be assisted by technologies such as cascading style sheets(CSS) and scripting languages like Javascript, Php, Python,Ruby,Perl etc. HTML elements are delineated by tags, written using angle brackets.

While i was already dealt with HTML, so it was not that much difficult for me to work with HTML.

CSS(cascading style sheets)

Like HTML i was dealt with CSS so it is pretty easy for me to with CSS.

I was used inline css (embedding the css in the same page as HTML) ie give style to the tag by using style property of the tag

syntax : <tag style = ‘style you need to give’></tag>

eg: <p style = ‘color: blue; margin-left: 20%;’>Snake game </p>

Javascript

Also in the case of javascript i was familiar with this so it is pretty handfull for me to work in js. ie, what i did was including script tag in the body section and write JS(javascript) code there.

```
<script>
//include javascript code here
</script>
```

Canvas

This is actually a new thing for me. So i referred MDN to understand that concept. The canvas is actually used to draw graphics using javascript, ie the canvas is containing the graphics that is specified in the script. Canvas is actually containing grid of elements(combination of x,y coordinates) if that concept is clear then everything becomes easy.
The basic syntax of canvas element is

```
    <canvas>
        //graphics
    </canvas>
```

TUTORIALS

consider the whole problem as a step by step process

step1 : Draw the canvas using the html

To draw a canvas use the <canvas> tag . We are going to use a canvas with these inline css properties

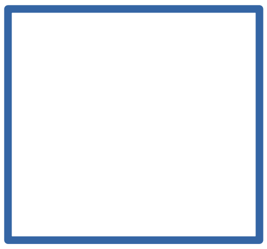
HTML

```
<body>

<canvas id = "mycanvas" width = "1008px" height ="504px" style = "padding: 1px;margin-left: 10%;border:5px solid #000000;color: blue;background-color: turquoise">

</body>
```

JS

```
var ele = document.getElementById('mycanvas');
```

This will draw a simple canvas with border 5px



Step 2: Drawing on the canvas

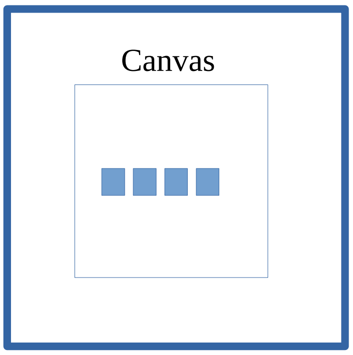
```
var ele = document.getElementById('mycanvas');
var context = ele.getContext('2d');
```

Consider the snake body part as piece of rectangles so we just need to draw rectangles inside the canvas

inside the canvas each position is indicated by some x,y coordinates so instead of giving the coordinates directly store each coordinates inside an array this will make the job easy for you. What i was done is to store x coordinates inside an array and store y coordinates in to an another array

```
if(context){
context.fillStyle = 'black'
context.fillRect(array_x[0],array_y[0],20,20);
context.fillStyle = 'red'
context.fillRect(array_x[1],array_y[1],20,20);
context.fillRect(array_x[2],array_y[2],20,20);
context.fillRect(array_x[3],array_y[3],20,20);
}
}
```

In this array_x indicating the x coordinates and array_y is indicating the y coordinates.



After this we will get something like this.

Step3 : Understanding the key movements

In this step find the arrow key movements . This is mainly to move the snake to a specified position.

JS

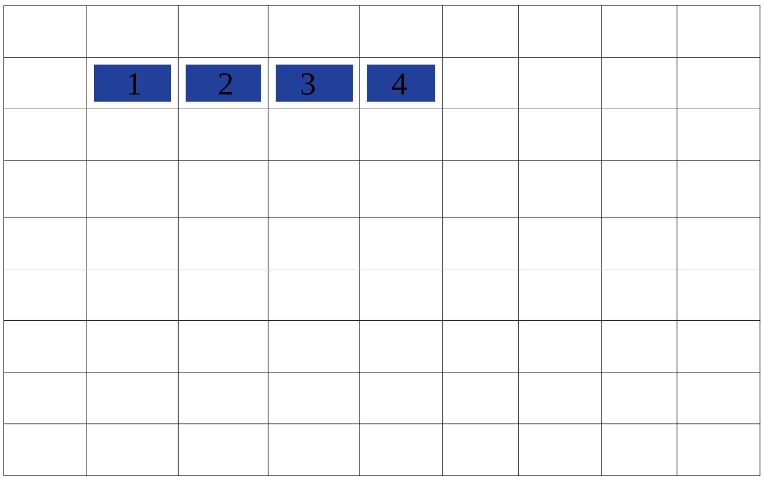
```
document.onkeydown = keyMovements;
function keyMovements(e){
e = e || window.event;
if (e.keyCode == '37'){
console.log("left arrow key is pressed")
}
else if (e.keyCode == '38'){
console.log("up arrow key is presssed")
}
else if(e.keyCode == '39'){
console.log("right arrow key is pressed")
}
else if (e.keyCode == '40'){
console.log("down arrow key is pressed")
}
```

In each arrow key press it will show that which arrow key is pressed in the web console.

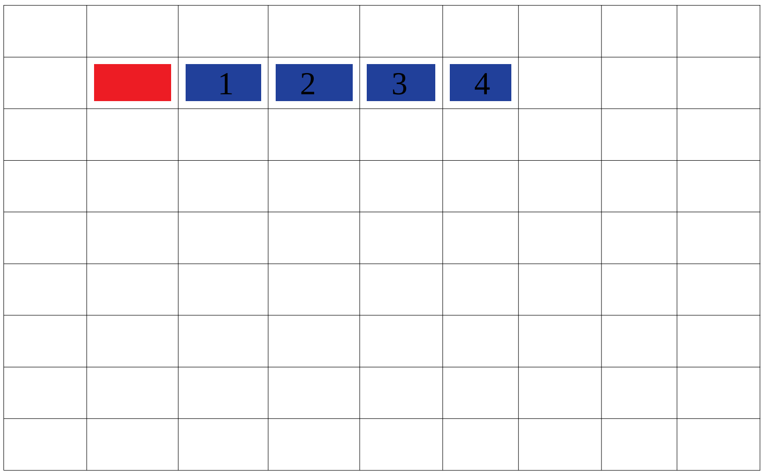
<u>Step 4</u> : Snake movement

In this step move the snake to any position on a specified key press. This is the point where the snake movement logic comes.

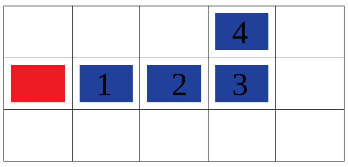
Consider this as the initial stage with 4 snake cells.



In the right movement the previous 1st element needs to deleted and new element needs to be created at the next cell.

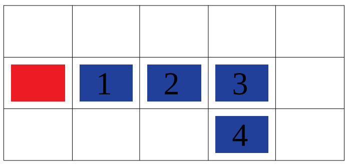


UP



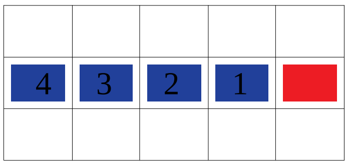
In the up movement the previous 1st cell needs to be deleted and new cell is created above the 3rd cell.

DOWN



In the down movement the previous 1st cell needs to be deleted and new cell is created below the 3rd cell.

LEFT



Since we are usign array, finding the positions of the snake is not that much difficult.

Consider the up movement in this.

```
<script>
if (e.keyCode == '38'){
up_move = setInterval(upMovement,100);
}
function upMovement(){
        context.clearRect(array_x[array_x.length-1],array_y[array_y.length-1],20,20)
        for(var i=array_y.length-1;i>=0;i--){
                array_y[i] = array_y[i-1]
                array_x[i] = array_x[i-1]
        }
        array_x[0] = array_x[1]
        array_y[0] = array_y[1]-22;
        for(var i=0;i<array_x.length;i++){
                context.fillStyle = 'red'
                context.fillRect(array_x[i],array_y[i],20,20)
        }
return;
}

</script>
```

In this when the up arrow key is pressed it will call an setInterval method

ie the setInterval method calls a function at specified intervals . The setInterval method is continue calling the function until the clearInterval method is called.

In this case when the up arrow key is pressed it will trigger upMovement method by setInterval in each 100 milliseconds.
Ie in the upMovement method first it will delete the last element , swap remaining elements to one position down and add the new element to the first position of the array.
In case of upmovement the y coordinate is decreased and the x is constant

```
array_y[0] = array_y[1]-22;
```

vice versa in other positions there is only a smaller differents in the code.

down movement : The y coordiante is increased and x is constant.
left movement   : The x coordinate is decreased and y is constant.
right movement : The x coordinate is increased and y is constant.

<u>Step 5</u> : Food eating and snake length increases

In this step need to create random food elements and and when the snake eats the food the length of the snake increased by one position at the tail.

(i)<u>Random food generating</u>

```
function food_eating(){
var x = Math.floor(Math.random()*(30-1+1))+1;
var y = Math.floor(Math.random()*(20-1+1))+1;
random_x = 22*x+8;
random_y = 22*y+-4;
food_check();
context.fillStyle = 'green'
context.fillRect(random_x,random_y,20,20);
}
function food_check(){
for(i=0;i<=array_x.length;i++){
if (random_x == array_x[i]){
if (random_y == array_y[i]){
food_eating();
}
}
}
}
```

In this step the food is generated at random locations inside the canvas region. And also it checks whether the food is generating at the same location of snake body if so generate another food which is not in the snake body part.

(ii) <u>Snake lenght increases</u> :

```
if(array_x[0]==random_x && array_y[0]==random_y){
array_x.length = array_x.length+1;
array_y.length = array_y.length+1;
food_eating();
}
```

In this step it will check the first element of the array is equal to the random food element coordinates, ie the snake head touches the food. If so the length of the array is increased by one.

<u>Step 6</u> : snake collisions

(i) <u>snake body collisions</u>:

While the length of the increases and when the snake head touches its body the snakes will dies.

```
function snake_body_collision(){
  for(i=1;i<=array_y.length;i++){
    if (array_y[0] == array_y[i]){
```

```
    if (array_x[0] == array_x[i]){
    window.alert('collision detected')
    location.reload();
  }
}
}
}
```

In this it will check the first element of the array is already containing in the array if so means that the head is touched its body so the game will end here.

(ii) <u>Boundary collisions</u> :

when the snake touches the canvas left, right, down, up regions the snake dies.

Ie in the up movement case

```
if (array_y[0] == -4){
window.alert("The game ends here")
location.reload();
}
```

Same in case of left,right and down regions only the boundary value will change.