Topic	BREAKING THE ROPE	
Class Description	**The student learns how to create a function to drop the fruit from the rope. The student will then create a bunny sprite and add a background image.**	
Class	**C30**	
Class time	**45 mins**	
Goal	• Drop the fruit by cutting the rope • Create the body for the rabbit. • Add background and rabbit image.	

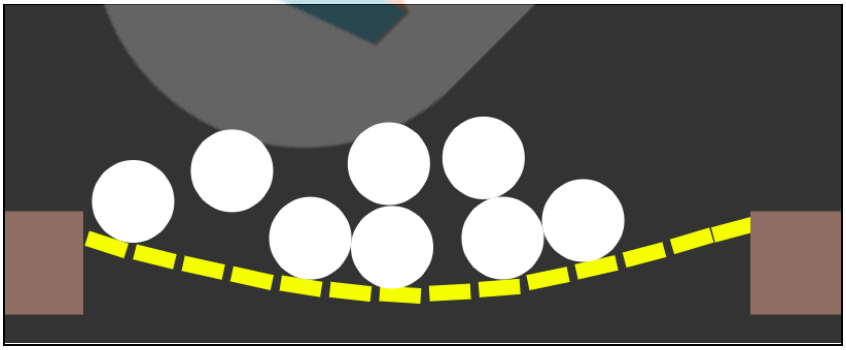
Resources Required	• Teacher Resources
	  ○ VS Code Editor
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	• Student Resources
	  ○ VS Code Editor
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen	

Class structure	**Warm-Up - Slide show option** **Teacher-Led Activity** **Student-Led Activity** **Wrap-Up - Slide show option**	**10 mins** **10 mins** **20 mins** **5 mins**

WARM-UP SESSION - 10 mins

Teacher starts slideshow  **from slides 1 to 11**
Refer to speaker notes and follow the instructions on each slide.

Teacher Action	Student Action

Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? ***Run the presentation from slide 1 to slide 3.*** **The following are the warm-up session deliverables:**	**ESR**: Hi, thanks, Yes I am excited about it! Click on the slide show tab and present the slides
QnA Session	
Question	**Answer**
Select the correct option to create a **bridge** using a Bridge class and **jointPoint** using **Base** Class. A. B. C. D.	A
Select the correct option to pass **bridge.body** and **jointPoint** in **Matter.Composite.add()**. 	A

A. Matter.Composite.add(bridge.body, jointPoint); B. Matter.Composite.add(jointPoint); C. Matter.Composite.add(jointPoint, bridge.body); D. Matter.Composite.add(bridge.body);	
Continue the warm-up session	
Activity details	**Solution/Guidelines**
Run the presentation from slide 4 to slide 11 to set the problem statement. **The following are the warm-up session deliverables:** ● Revise how to load images	Narrate the slides by using hand gestures and voice modulation methods to bring in more interest in students.
Teacher ends slideshow 	
TEACHER-LED ACTIVITY - 10 mins	
Teacher Initiates Screen Share	
CHALLENGE ● **Adding the background and fruit image.**	
Teacher Action	**Student Action**
Teacher-led Activity 1 In the previous class, we created our rope and hanged the fruit body with it. In this class, we are going to write the code to remove the fruit body from the rope, when the user will press the “cut” button and the fruit will then fall. We will also create the body for the bunny and add images	*The teacher downloads and opens the Teacher Activity 1 code in the VS Code editor.*

for it as well as for the background. First, let's add all the images. We need images for the background, our bunny, and the cut button. Images are present in the folder. We now need to load the images in our code. Can you tell me how we're going to do this? Very good, let's write the **preload()** function. We will write this code in the **sketch.js** file. We load the assets in the **preload()** function because we want to load all the images before our main code begins, and the **preload()** function runs at first, all the other functions and instructions are executed after the preload function. First we need to declare 3 variables for images as, **var bg_img, fruit_img, bunny_img;** Define the **preload()** function, and now we will add the images for the background, fruit, and bunny. Load the image using the **loadImage()** function and pass the image path as the parameter within the **loadImage()** function. In this way, images will be loaded, and we will assign them to the variables we created earlier.	**ESR**: By using a **loadImage()** and the **preload()** function.

```
function preload()
{
  bg_img = loadImage('background.png');
  food = loadImage('melon.png');
  rabbit = loadImage('Rabbit-01.png');
}
```

Once we have the images, now we can set them at their respective places.

But before doing that, we need to add one more instruction in the **setup()** function.

Which is to set the image mode as the center.
With this, while displaying the image on the canvas, we will have the center point of the images.

By default, when we create the image on the canvas, it takes the top left corner as the origin point, but here we are changing it to the center, so now when we draw the image on the canvas, we will specify its center position as the x and y coordinates.

To do that we need to write the **imageMode()** function and add **CENTER** as the parameter in the function.
We have loaded the images now and let's display them on the canvas.

First is the background image.

To display the image on the canvas, we will use the **image()** function. This function will take the image's x, y position, and width and height of the image as arguments.

The image is stored in the **bg_img** variable, x and y position we will specify as width/2 and height/2 because we set the **imageMode** as **CENTER**. Then followed by the

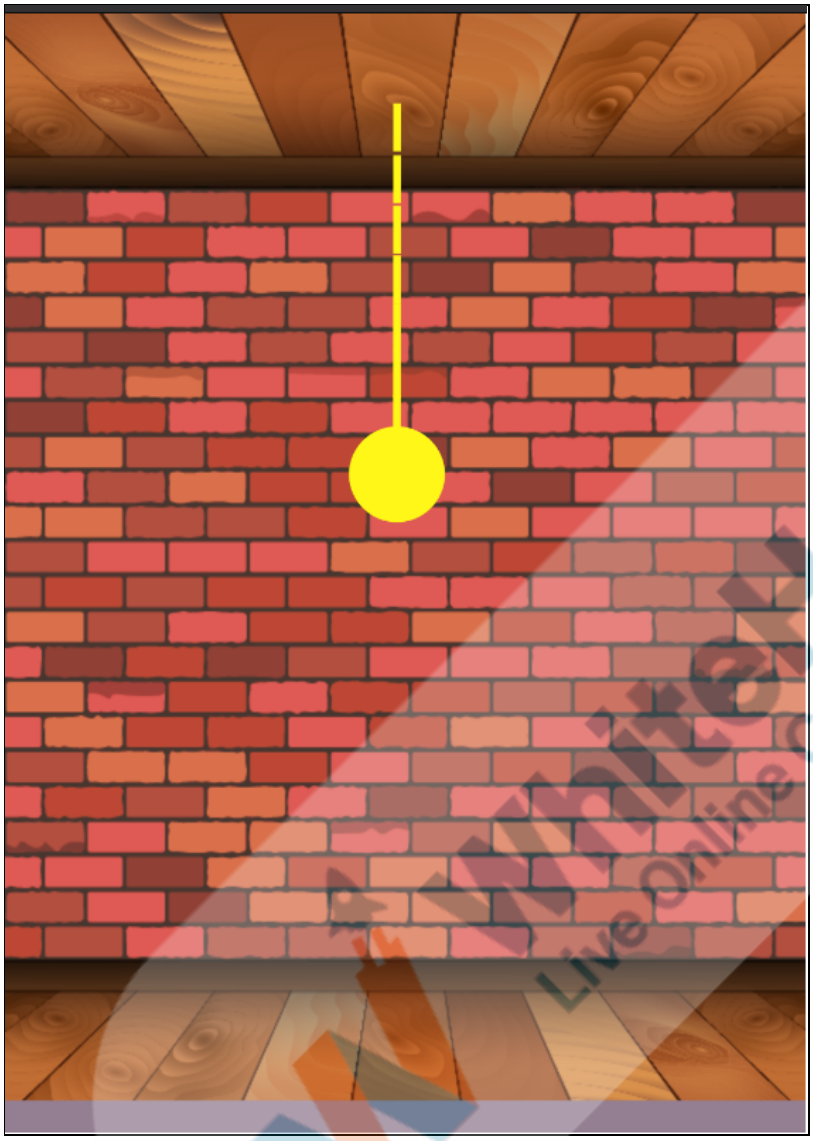
size of the image as the size of the canvas, which is **500** and **700** respectively.

In the **draw()** function, write the **image()** function and pass the corresponding arguments.

```
function setup()
{
  createCanvas(500,700);
  frameRate(80);
  engine = Engine.create();
  world = engine.world;
  ground = new Ground(200,680,600,20);

  rope = new Rope(7,{x:245,y:30});
  fruit = Bodies.circle(300,300,20);
  Matter.Composite.add(rope.body,fruit);

  fruit_con = new Link(rope,fruit);

  rectMode(CENTER);
  ellipseMode(RADIUS);
  textSize(50);

  imageMode(CENTER);

}
```

The teacher runs the code.

When you run by clicking on the **GoLive** button, the code will show the background in our game.



Now let's add the fruit image.

We have already created the fruit body and right now we are creating a circle using the positions of the fruit body.

We will add the image of the watermelon to the same location, but we need to remove the circle.

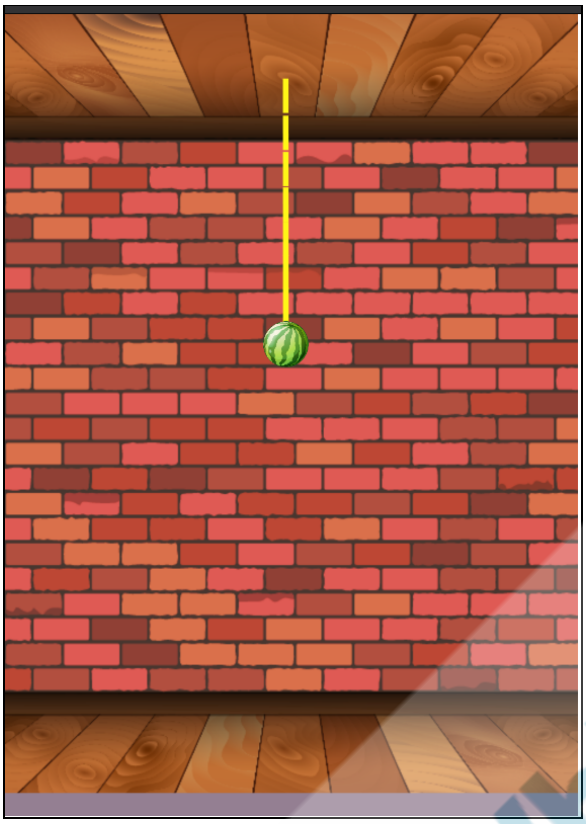
We will also make the fruit image look a little bigger and

real, so we will keep the width and height of the watermelon image as **60**,**60**.

Here **Engine.update(engine)** will update the output.

```
function draw()
{
  background(51);
  image(bg_img,width/2,height/2,500,700);
  ground.show();
  rope.show();

  image(fruit_img,fruit.position.x,fruit.position.y,60,60);

  Engine.update(engine);
}
```

Output:



We have added the background and fruit image. Now it's your turn to add a button and functions to cut the rope and drop the fruit. What do you think? How can we create a button on the canvas? Ok, let's find out. Please share your screen with me.	**ESR:** Varied

Teacher starts slideshow : Slide 12 to 18

Run the presentation slide to set the student activity context.	
• **Create a function to add a button and a function to drop the fruit** • **Add the bunny sprite**	

Teacher ends slideshow 	
Teacher Stops Screen Share	
STUDENT-LED ACTIVITY - 20mins	
• **Ask Student to press ESC key to come back to panel** • **Guide Student to start Screen Share** • **Teacher gets into Fullscreen**	
ACTIVITY • **Create the button and function to drop the fruit.** • **Add the bunny sprite.**	
Teacher Action	**Student Action**
Student-led Activity-1 We have added the images for the fruit and the background. Now let's create the Sprite for the bunny. Fruit, ground, and rope are physics bodies, which are using the physics library **matter.js**. For the bunny, we will not create a physics body, because we only need it to play its animation and detect the collision with fruit.	*The student downloads Student Activity 1 code and opens it in the VS Code editor.*

That is why we are going to use the **p5.play library**, which we used to make the T-rex game.

The library is already added in the code you have downloaded just now, so we can directly move on to create the sprite.

To create a sprite, first, declare a variable for that,
var bunny;

We are writing this code in the **sketch.js** file.

Now in the **setup()** function we create the sprite using the **createSprite()** function.
In this function, we need to pass the x, y positions, and the width and height of the sprite.
We need to add the image to the bunny as well, for that we will use **bunny.addImage()** function.

We will also set the image for the bunny sprite and set the scale of the sprite as **0.2** because the image we have is a lot bigger and may not fit in our canvas, so we have to scale it down.

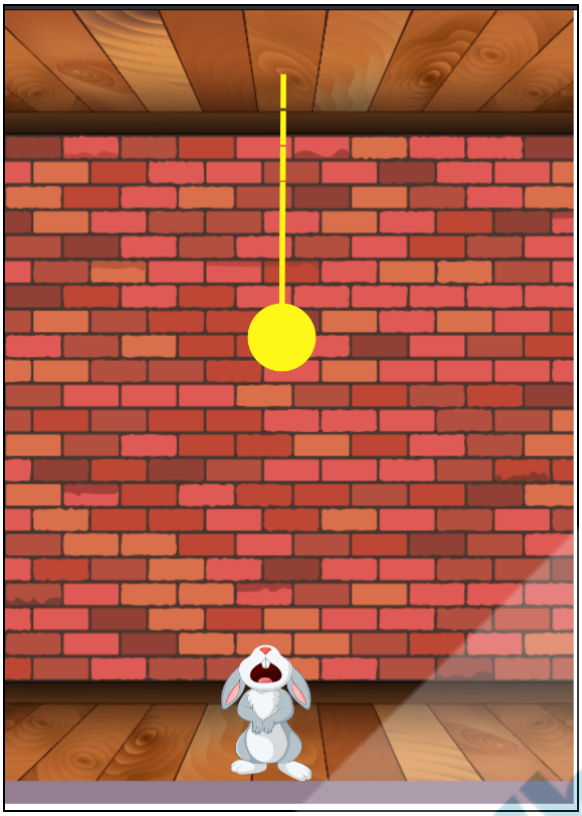
```
function setup() {
  createCanvas(500,700);
  frameRate(80);
  engine = Engine.create();
  world = engine.world;
  bunny = createSprite(250,650,100,100);
  bunny.addImage(bunny_img);
  bunny.scale = 0.2;
```

If you run the code, you won't be able to see the bunny on the screen.

Can you tell me why? Let's add the **drawSprite()** within the **draw()** function and run our code.	**ESR:** We need to draw the sprite using the **drawSprite()** function.

```
function draw()
{
  background(51);
  image(bg_img,width/2,height/2,500,700);
  ground.show();
  rope.show();

  image(fruit_img,fruit.position.x,fruit.position.y,60,60);
  Engine.update(engine);

  drawSprites();
}
```

Output:



Now we need to create a function to drop the melon and break the rope.	
We have two things to perform: • Cut the rope from the top point where it is connected. • Cut the fruit from the rope. To cut the rope from the top, we are going to call a function from the rope class, which is called **break()**. It will release the rope and remove it from the scene. Next, to detach the fruit from the rope, we need to delete the constraint between the rope and fruit by removing that constraint from the world.	

We will create this function in the **Link** class so that we can call it using the link object. In this, we will use the **World.remove()** function, this will remove the constraint from the world. To call the **detach()** & **break()** function we will create a new function in the **sketch.js** file named **drop()** which we will call(execute) by a button press. We will also make the fruit constraint as **null**. So that it does not affect the fruit.	

```
class Link{
    constructor(bodyA,bodyB)
    {
      var lastlink = bodyA.body.bodies.length-2;
     this.link = Constraint.create(
        {
          bodyA:bodyA.body.bodies[lastlink],
          pointA:{x:0,y:0},
          bodyB:bodyB,
          pointB:{x:0,y:0},
          length:-10,
          stiffness:0.01
        });
        World.add(engine.world,this.link);
    }

    detach()
    {
      World.remove(engine.world,this.link);

    }
}
```

Now we will define the **drop()** function, where we're going to break the rope using the **rope.break()** function, and we remove the fruit constraint by using the **fruit_con.detach()** function and making **fruit_con** as null.

```
function drop()
{
  rope.break();
  fruit_con.detach();
  fruit_con = null;
}
```

We have created our function, now let's quickly create a button and add this function with that button, so that we can drop our fruit.

In **p5.js** we have in-built functions to create buttons on the screen, and we have already done that exercise in multiple classes before as well.

Where we used the **createButton()** function.
But that creates a simple button on the canvas, and it does not look very visually appealing.
So, now we will use a different function called **createImg()**.

This takes an image as a parameter and makes that image work as a button.

First, define a variable as a **var** button.

Then in the **setup()** function create the image button using the **createImg()** function and pass the image we want to display on this button.

We also need to specify the size and position of the button.

Then finally, we will add the function with the help of the

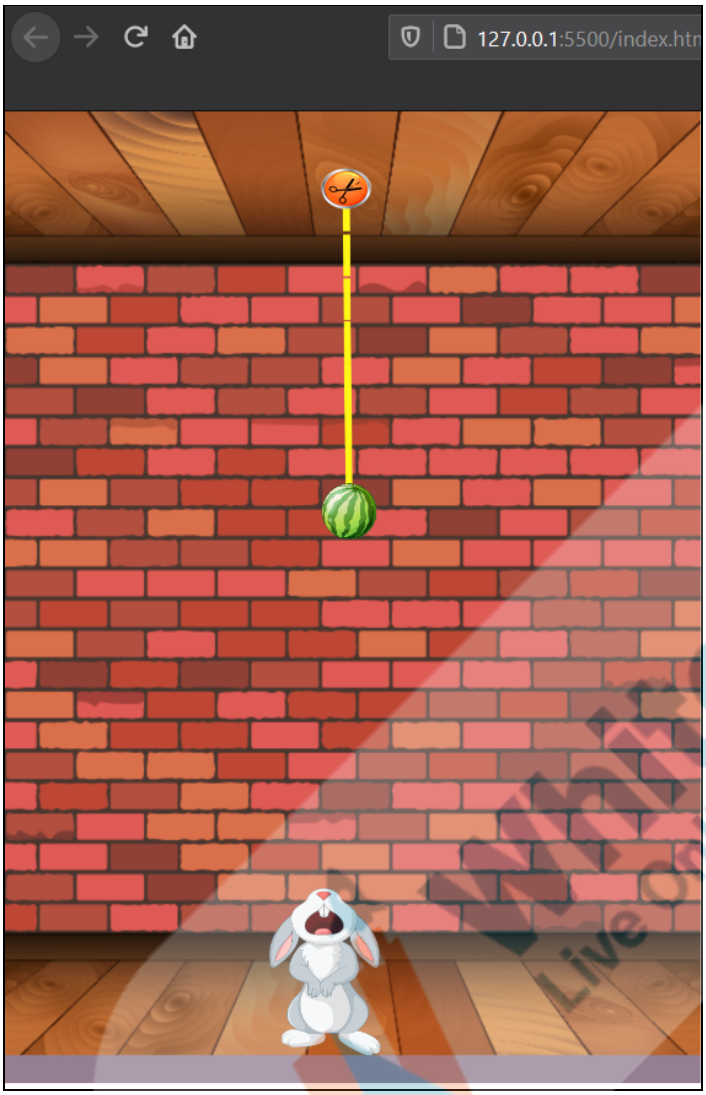
mouseClicked() function.

In the **mouseClicked()** function, we are passing the function which we want to execute when we click the button.

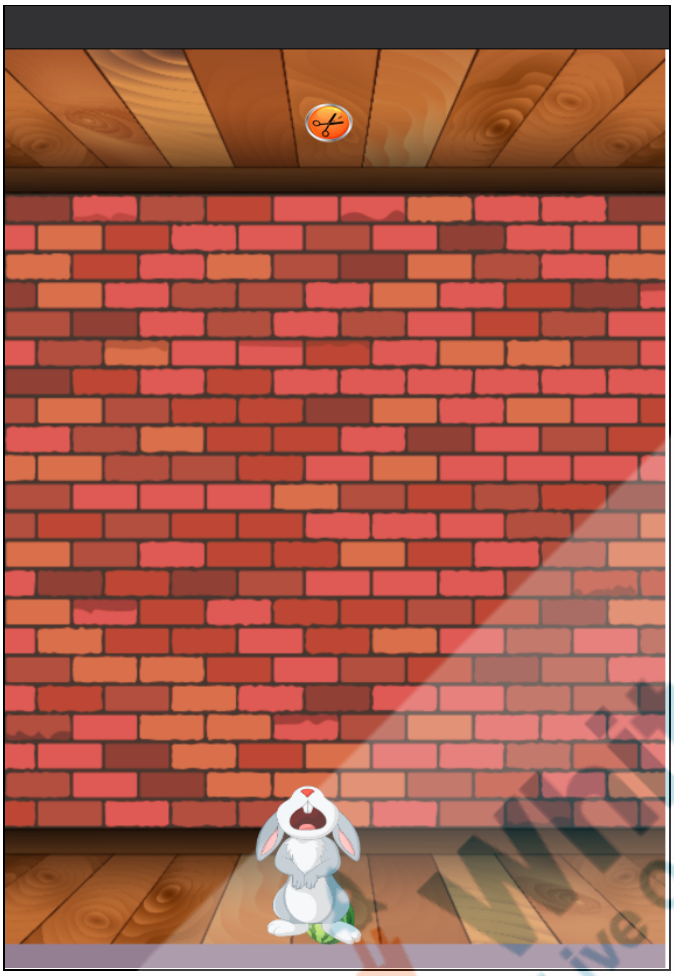
Let's run the code and see if it is working.

```
button = createImg('cut_btn.png');
button.position(220,30);
button.size(50,50);
button.mouseClicked(drop);
```

Output:



We have our button, let's click on the button and see if our fruit drops or not.



That's great, our code is working, and the fruit drops properly. You can try a couple of things. Refresh the output page and click on the button when the rope is swinging, you will see that fruit drops at different locations.	
We got the basic structure of the game. In the next class, we are going to add animation to our bunny, and we will detect the collision of the fruit with the bunny to play different animations.	

Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 5 Mins	
Teacher starts slideshow  **from slide 19 to slide 28**	
Activity details	**Solution/Guidelines**
Run the presentation from slide 19 to slide 28. **Following are the wrap-up session deliverables:** • **Explain the facts and trivias** • **Next class challenge** • **Project for the day** • **Additional Activity**	Guide the student to develop the project and share it with us
Teacher ends slideshow 	
Quiz time - Click on in-class quiz	
Question	**Answer**
Why do we use the imageMode(CENTER) command? A. Because by default, when we create the image on the canvas, it takes the top left corner as the origin point. B. To draw the image at the center of the canvas. C. To draw the image at the top left corner of the canvas. D. None of the above.	**A**
Which of the following commands is used to remove the	**C**

body from the world? A. World.remove(body,world) B. World.remove(world,body) C. World.remove(body) D. body.remove(world)		
Which p5.js in-built function is used to create buttons on the screen? A. create.button() B. createButton() C. button.create() D. Createbutton()		**B**
End the quiz panel		
	You get hats off.	Make sure you have given at least 2 Hats Off during the class for:   
*** This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.** **Project Overview** **CRUSH THE ZOMBIES - 2**		*The student engages with the teacher over the project.*

Goal of the Project:

In this project, we will create a zombie and add an image to it along with the movement. We'll also add the images to the stones, bridge, and background. We'll also add a button which when pressed the bridge will collapse releasing the stones on the zombie.

Story:

A far, far away village is always troubled by the zombie. The only way to kill the zombie is to drop a stone on its head. You have mostly seen that the zombie travels under the bridge to get to the village, so you plan to stack the bridge with the stones and drop it on the zombie when it comes under the bridge.

I am very excited to see your project solution and I know you will do really well.

Bye Bye!

Links:

Activity	Description	Link
Teacher Activity 1	Boilerplate code	https://github.com/pro-whitehatjr/C-30-TA1
Student Activity 1	Boilerplate code	https://github.com/pro-whitehatjr/C-30SA1
Reference Code	Complete code for reference	https://github.com/pro-whitehatjr/C-30_complete
Project Solution	Zombie 2	https://github.com/whitehatjr/zombie-crush-2
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC-PRO-V3-C30-withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/a3eedd56-15c1-43db-9dc1-89df71fab1e0.pdf
Project Solution	Crush The Zombies-2	https://github.com/whitehatjr/zombie-crush-2