Given below are the world functions.

Tiles at the bottom of the Alice window:

- Do in order
- Do together
- If/Else
- Loop
- While
- For all in order
- For all together
- Wait
- Print

Tiles at the bottom of the Alice window.
Events:

- When the world starts
- When a key is typed
- When the mouse is clicked on something
- While something is true
- When a variable changes
- Let the mouse move <objects>
- Let the arrow keys move <subject>
- Let the mouse move the camera
- Let the mouse orient the camera

Not shown: When something is true

Format of instructions:

- If true
  - Do Nothing
- Else
  - Do Nothing

- Loop 5 times
  - Do Nothing

- While true
  - Do Nothing

- For all world:horses, one of item_from_horses at a time
  - Do Nothing

- For all world:horses, every of item_from_horses together
  - Do Nothing
Chicken properties and methods:

<table>
<thead>
<tr>
<th>Chicken's details</th>
<th>Chicken's details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Properties</strong></td>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td>create new variable</td>
<td>create new method</td>
</tr>
<tr>
<td>capture pose</td>
<td>Chicken move</td>
</tr>
<tr>
<td>color =</td>
<td>Chicken turn</td>
</tr>
<tr>
<td>opacity = 1 (100%)</td>
<td>Chicken roll</td>
</tr>
<tr>
<td>vehicle = world</td>
<td>Chicken resize</td>
</tr>
<tr>
<td>skin texture = Chicken.TextureMap</td>
<td>Chicken say</td>
</tr>
<tr>
<td>fillingStyle = solid</td>
<td>Chicken think</td>
</tr>
<tr>
<td>pointOfView = position: 0, 0, -0.1; orientation: (0, 0, 0) 1</td>
<td>Chicken play sound</td>
</tr>
<tr>
<td>isShowing = true</td>
<td>Chicken move to</td>
</tr>
<tr>
<td></td>
<td>Chicken move toward</td>
</tr>
<tr>
<td></td>
<td>Chicken move away from</td>
</tr>
<tr>
<td></td>
<td>Chicken orient to</td>
</tr>
<tr>
<td></td>
<td>Chicken turn to face</td>
</tr>
<tr>
<td></td>
<td>Chicken point at</td>
</tr>
<tr>
<td></td>
<td>Chicken set point of view to</td>
</tr>
<tr>
<td></td>
<td>Chicken set pose</td>
</tr>
<tr>
<td></td>
<td>Chicken stand up</td>
</tr>
<tr>
<td></td>
<td>Chicken move at speed</td>
</tr>
<tr>
<td></td>
<td>Chicken turn at speed</td>
</tr>
<tr>
<td></td>
<td>Chicken roll at speed</td>
</tr>
<tr>
<td></td>
<td>Chicken constrain to face</td>
</tr>
<tr>
<td></td>
<td>Chicken constrain to point at</td>
</tr>
</tbody>
</table>
Given below are the chicken functions.

<table>
<thead>
<tr>
<th>Proximity</th>
<th>Spatial Relation</th>
<th>Point of View</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicken is within threshold of object</td>
<td>Chicken is shorter than</td>
<td>Chicken's point of view</td>
<td>Chicken's current pose</td>
</tr>
<tr>
<td>Chicken is at least threshold away from object</td>
<td>Chicken is taller than</td>
<td>Chicken's position</td>
<td>Chicken's part named key</td>
</tr>
<tr>
<td>Chicken distance to</td>
<td>Chicken is to the left of</td>
<td>Chicken's quaternion</td>
<td></td>
</tr>
<tr>
<td>Chicken distance to the left of</td>
<td>Chicken is to the right of</td>
<td></td>
<td>Chicken's variable named variableName of type valueClass</td>
</tr>
<tr>
<td>Chicken distance above</td>
<td>Chicken is above</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chicken distance below</td>
<td>Chicken is below</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chicken distance in front of</td>
<td>Chicken is in front of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chicken distance behind</td>
<td>Chicken is behind</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Array Visualization special methods and functions

ArrayVisualization's details

1. let ArrayVisualization [ index ] = item
2. ArrayVisualization's size
1. (12 pts) Consider the following snapshot of an Alice world in which there are three objects which are (from left to right) beachChair, penguin, and chair. The beachChair and penguin are 2 meters apart. The penguin and chair are 2 meters apart. Imagine the three of them are standing on an invisible line. Note that the method penguin.wing_flap times=2 has the penguin flap his wings twice.

A) List the events in this world.

B) List the event handlers in this world.

C) Explain what happens in the world when the Play button is pressed.

D) About five minutes after the Play button is pressed, the user clicks on the penguin and moves the penguin back to his original position. Explain what happens in the world.
2. (4 pts) Consider the following array of animals shown below as a picture and as a property, and the following code. Also shown in the picture are an ArrayVisualization and an ObjectVisualization.

Here is the code:

```plaintext
Loop [123] index from 2 up to (but not including) arrayVisualization's size incrementing by 1

let objectVisualization = the value at arrayVisualization [ index ] more...

let arrayVisualization [ index ] = the value at arrayVisualization [ ( index - 2 ) ]

let arrayVisualization [ ( index - 2 ) ] = the value of objectVisualization more...

Loop [123] index from 0 up to (but not including) arrayVisualization's size incrementing by 1

print the value at arrayVisualization [ index ]
```

Remember that when an object is printed, its name is printed. What output is printed when this code is executed?
3. (8 pts) Consider the following function called `world.mystery`. Assume a list called `zooAnimals` has been created and filled with objects.

A. What type of value does the function `world.mystery` return?

B. What is `someColor`: a local variable, a class variable or a parameter?

C. What is `total`: a local variable, a class variable or a parameter?

D. What is `value`: a local variable, a class variable or a parameter?

E. Explain what the function `world.mystery` does.

F. If the second if statement that starts with “If total > value” is moved and placed after the for loop, will Mystery always return the same value as the original code would have?
4. (4 pts) Consider the following Alice code.

A. How many times does cleopatra say “Help” when this code executes?
B. How many times does the mummy chase cleopatra around the palmTree?
C. How many times does cleopatra jump up and down when this code executes?
5. (8 pts) Consider the following recursive function `world.mystery` whose lines are numbered.

A. Which line(s) of code have recursion?

B. Explain where the “way out” of the recursion is.

C. What is the output of the following code?

```
print world.mystery input = 6 < factor = 4
print world.mystery input = 4 < factor = 6
print world.mystery input = 11 < factor = 2
print world.mystery input = 20 < factor = 10
```
6. (10 pts) Consider the following Alice world, which includes a 3DText called score, a baseball, a beachball and an alienOnWheels. There is an event to click on any Object to move it. In the world a game is played in which the balls move around randomly. The user uses the mouse to move the alienOnWheels around in the world. The user gets points if the alienOnWheels is close to either ball. The balls change colors. In the picture below the top ball is the beachball and it is currently colored red. The ball below it is the baseball.

![Image of the world with score, baseball, beachball, and alienOnWheels]

Answer questions about this world on the next page.

A. Write the method score.increase that has one object parameter named item. The score is increased by 3 if the item’s color is red, and otherwise the score is increased by 1. In either case, the new score value must be displayed in the world. In the picture on the previous page the score is displayed as 0.0.

![Image of the score increase method]

B. Give two events for the following. For one event, whenever the alienOnWheels is close to the baseball, then the score should be increased by 1 (by 3 if the baseball is red). There should be a similar event whenever the alienOnWheels is close to the beachball. In writing these events, you should call the method you wrote in Part A.
7. (8 pts) Write the function `world.numberInRange` that has two number parameters, `min` and `max`. This function returns the number of items in the array values that are in the range from `min` to `max` (inclusive). For example, the picture on the right below shows the array values. The first call on the left would print 8, as all 8 values are between 1 and 42. The second call on the left would print 5, as only five of the values (7, 15, 8, 10, 5) are between 5 and 15.

Write your code below. Your code should work regardless of the numbers and size of values. That is, your code should still work if numbers in `values` are changed, or new numbers are added to the array.

8. (20 pts) Assume all the chickens below are in a list called `chickens`.

A. Write the method `world.randomlyColor` with one parameter: `color`. This method randomly colors each chicken either the color parameter or the color white. For example, in the picture above, `world.randomlyColor color=Blue` might randomly color 3 of the chickens blue and the rest white. Complete the method below.
B. Write the function `world.ratioThisColor` that has one parameter `color` and returns the ratio of chickens in the list `chickens` that has this color. For example, if there were five blue chickens and three white chickens, then the call `world.ratioThisColor` with `color == Blue` would return 0.625 which is 5 divided by 8. Complete the function below.

```
world.ratioThisColor
```

C. Write the method `world.fixTheColoring` to do the following. Color the chickens randomly either white or blue. If more than half of the chickens are not blue, then randomly color the chickens either white or blue again. Keep repeating this until more than half of the chickens are blue. Finally print out a detailed message (in one line) stating the percentage of blue chickens. For example, in the picture below, there are 6 blue chickens (the shaded ones) and 2 white chickens, thus the ratio of blue chickens is 0.75. The output is below the picture stating “The percentage of blue chickens is 0.75.”

```
world.fixTheColoring
```

Complete the method below. You must call the function and method you wrote in parts A and B.

9. (14 pts) Consider a world with the array Visualization of people shown below and an iceSkater to the left of the array and a coach to the right of the array.
A) Write the function `World.lastOneShorter` that has one object parameter called `person`. This function returns the **index of the last** person in the array who is shorter than the parameter person. If no one is shorter than person, then return the index of the last person in the array.

```
world.lastOneShorter
```

B) Write the method `world.Greetings` that has one object parameter named `visitor`. This method has each person in the array say “Hello”. There is one exception. If there is someone in the array that is shorter than the parameter visitor, the last such shorter person in the array (the one on a higher index number, in the picture that would be the shortest person from the right side) says “Boo Hoo, I am shorter than you” instead of “Hello”.

For example, in the picture above, the call `Greetings visitor=iceSkater` results in each person saying Hello, except for AliceLiddell, the 5th item in the array is the last item in the array that is shorter than the iceSkater, she says “Boo Hoo, I am shorter than you.”

Write the method below. You must call the function you wrote in Part A.