

## CompSci 94, Fall 2013 Test 2 Solutions

1. A) There are four events:

When the world starts

While penguin distance to lemur  $> 0.9$

When lemur distance to penguin  $< 1$

When the mouse clicks on an object

B) There are four event handlers with one having multiple parts:

World.myFirstMethod

All parts of the BDE: penguin turn to face lemur, penguin move forward .5 meters, and penguin say Hello

World.something

The mouse moving an object

C) The penguin faces the lemur and repeatedly moves towards him until he is .5 meters from the lemur and he says hello. As the penguin reaches the lemur, the When event becomes true and the lemur turns to face the tortoise and moves close to it. The BDE is true again so the penguin moves over the the lemur and says hello. The When event is true again but the lemur is already close to and facing the tortoise.

D) A – if the tortoise is moved to the well. Nothing else happens.

B – if the lemur is moved to the well, then the penguin will move over to the lemur and say hello, the lemur will then move over to the tortoise and then the penguin will move over to the lemur's new position and say hello again.

C – if the penguin is moved to the well, then the penguin will move over to the lemur and say hello.

2. turtle

cat

penguin

tortoise

lemur

joey

3. A) Mystery returns a number

B) temp is a local variable

- C) creatures is a list of objects
  - D) something is a parameter
  - E) mystery returns the number of objects in the list creatures that are a distance less than 2 from the object something.
4. A) The troll says yikes 10 times.
  - B) The troll's club turns 150 times.
  - C) The dukePrince turns 50 times.
5. A. Score.increment

The image shows a Scratch script titled "score.increment" with a parameter "ball" of type "Obj". The script starts with a local variable "ranNumber" set to 1. It then uses a series of nested if-else statements to determine the score increase based on the "ball" parameter:

- If `ball == soccerBall`, `score.value` is set to `( score.value + 1 )`.
- Else, if `ball == basketball`, `score.value` is set to `( score.value + 2 )`.
- Else, if `ball == football`, `ranNumber` is set to a random number between 3 and 6 (integer only), and `score.value` is set to `( score.value + ranNumber )`.
- Else, `Do Nothing`.

Finally, the `score` variable is set to `score.value` as a string.

B) The score increase for the soccerBall and football are the same, the score is increased by 1 point for the soccerBall and a random integer 3, 4, or 5 for the football. The score is increased by four points for the basketball. By adding an additional event for the basketball, both events add 2 to the score when the basketball is clicked on.

6. Code is:



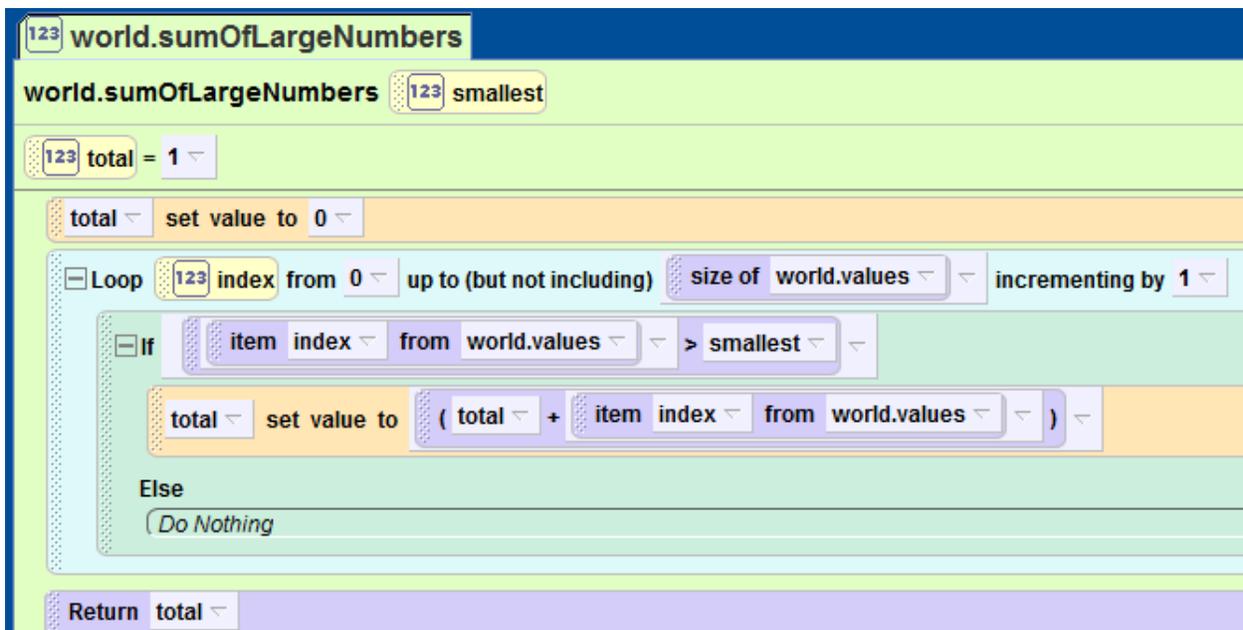
The image shows a Scratch code block titled "world.MakeAllButNumInvisible". It has a parameter "number" with the value "123". Below the parameter, it says "No variables". The code block contains a "While" loop with the condition "world.numberVisible > number". Inside the loop, there is a "world.makeInvisible" block.

Alternate code is:



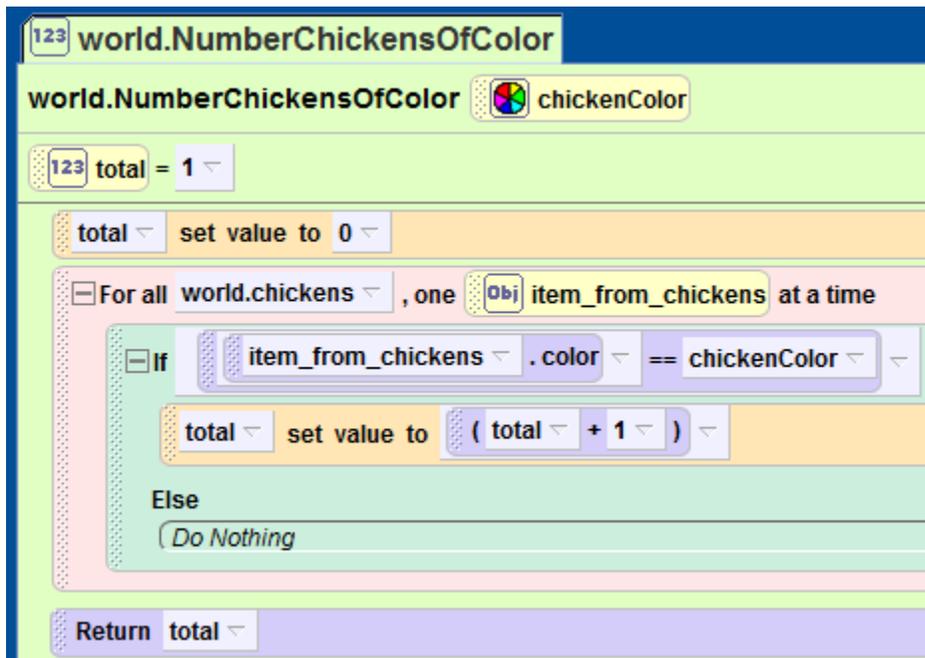
The image shows a Scratch code block titled "world.makeAllButNumInvisible2". It has a parameter "number" with the value "123". Below the parameter, it says "No variables". The code block contains a "Loop" block with the condition "( world.numberVisible - number ) < time". Inside the loop, there is a "world.makeInvisible" block.

7. Code is:



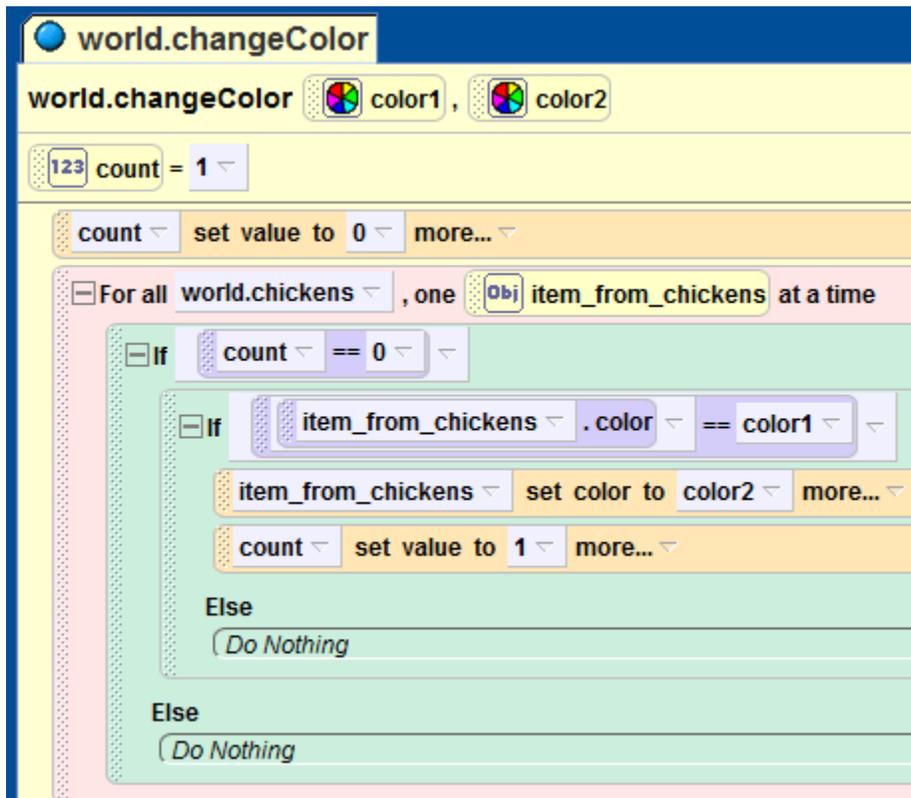
The image shows a Scratch code block titled "world.sumOfLargeNumbers". It has a parameter "smallest" with the value "123". The code block starts with a "total = 1" block. Then, there is a "total set value to 0" block. This is followed by a "Loop" block with the condition "index from 0 up to (but not including) size of world.values incrementing by 1". Inside the loop, there is an "If" block with the condition "item index from world.values > smallest". Inside the "If" block, there is a "total set value to ( total + item index from world.values )" block. Below the "If" block, there is an "Else" block with the text "Do Nothing". Finally, there is a "Return total" block.

8. A) Code is:



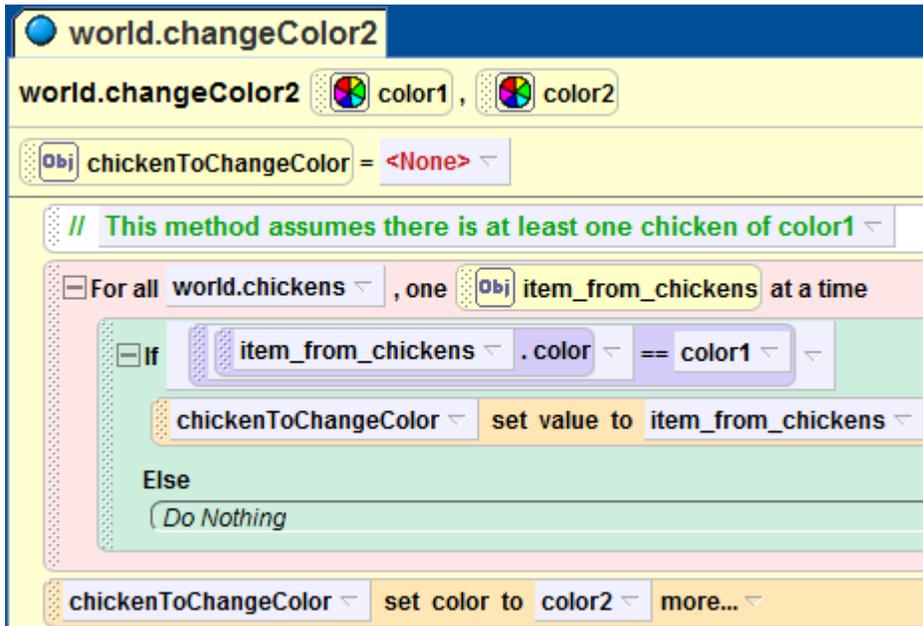
The image shows a Scratch code editor for a function named `world.NumberChickensOfColor`. The function takes a parameter `chickenColor` (represented by a color wheel icon). The code starts with a comment `total = 1`. The first block is `total` set value to `0`. This is followed by a `For all` loop over `world.chickens`, with one `Obj` `item_from_chickens` at a time. Inside the loop, there is an `If` block: `item_from_chickens` `.color` `==` `chickenColor`. If true, `total` is set value to `( total + 1 )`. The `Else` block contains `Do Nothing`. After the loop, the function returns `total`.

B) Code is:



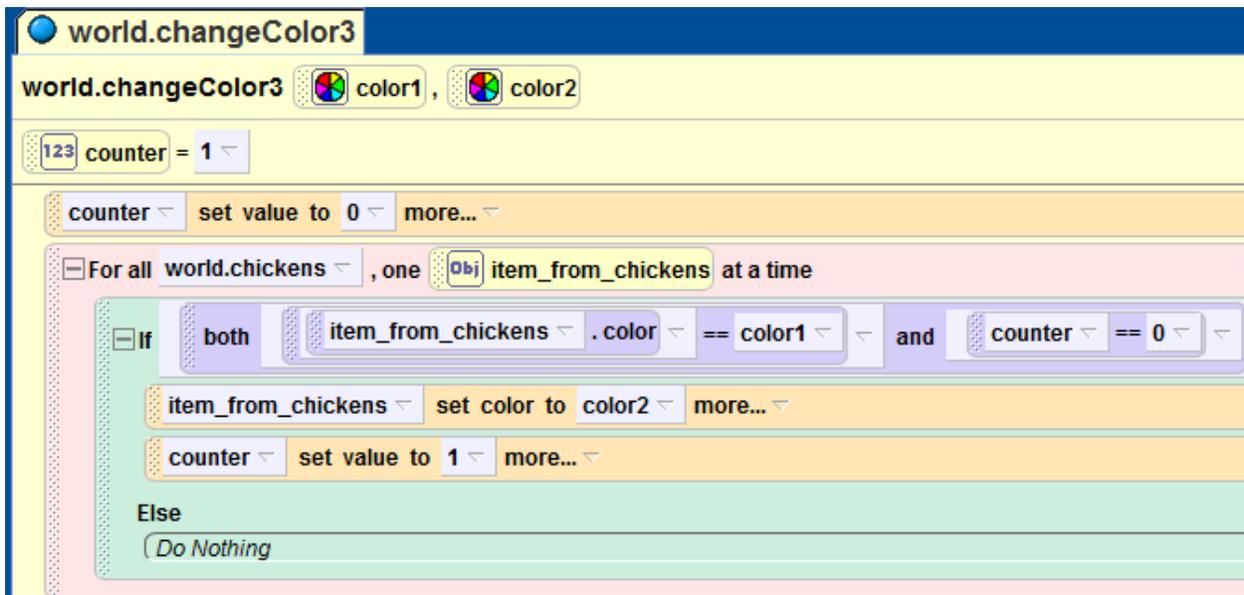
The image shows a Scratch code editor for a function named `world.changeColor`. The function takes two parameters, `color1` and `color2` (represented by color wheel icons). The code starts with a comment `count = 1`. The first block is `count` set value to `0` more... This is followed by a `For all` loop over `world.chickens`, with one `Obj` `item_from_chickens` at a time. Inside the loop, there is an `If` block: `count` `==` `0`. If true, there is another `If` block: `item_from_chickens` `.color` `==` `color1`. If true, `item_from_chickens` set color to `color2` more... and `count` set value to `1` more... The `Else` block contains `Do Nothing`. The outer `If` block's `Else` block also contains `Do Nothing`.

Alternate Solution:



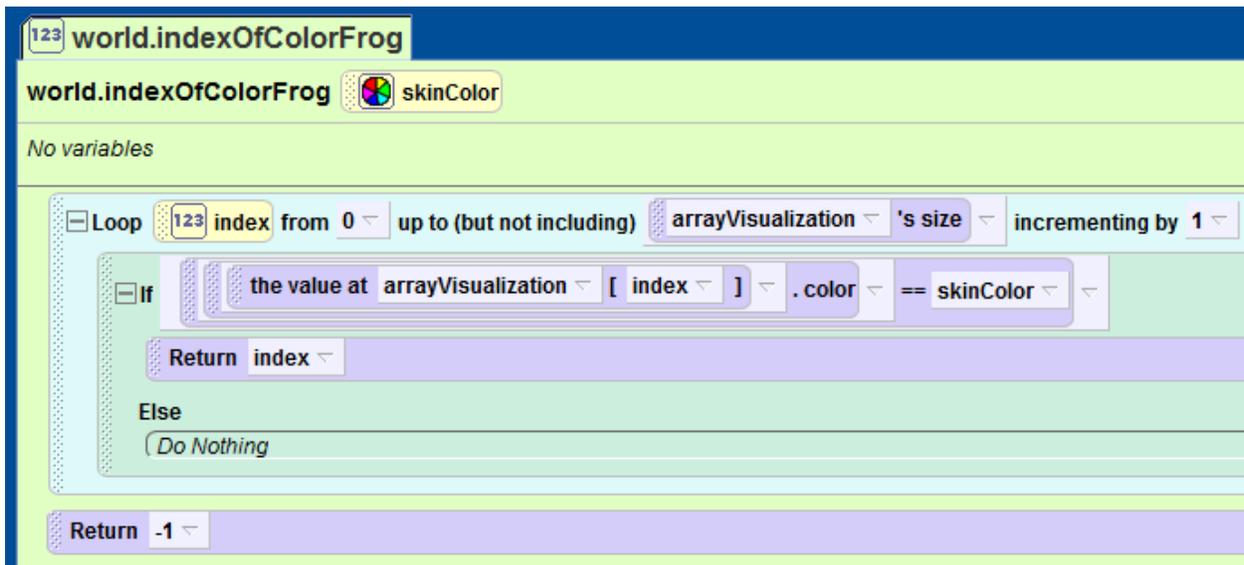
```
world.changeColor2 color1, color2
Obj chickenToChangeColor = <None>
// This method assumes there is at least one chicken of color1
For all world.chickens, one Obj item_from_chickens at a time
  If item_from_chickens.color == color1
    chickenToChangeColor set value to item_from_chickens
  Else
    Do Nothing
chickenToChangeColor set color to color2 more...
```

Alternate Solution:



```
world.changeColor3 color1, color2
123 counter = 1
counter set value to 0 more...
For all world.chickens, one Obj item_from_chickens at a time
  If both item_from_chickens.color == color1 and counter == 0
    item_from_chickens set color to color2 more...
    counter set value to 1 more...
  Else
    Do Nothing
```

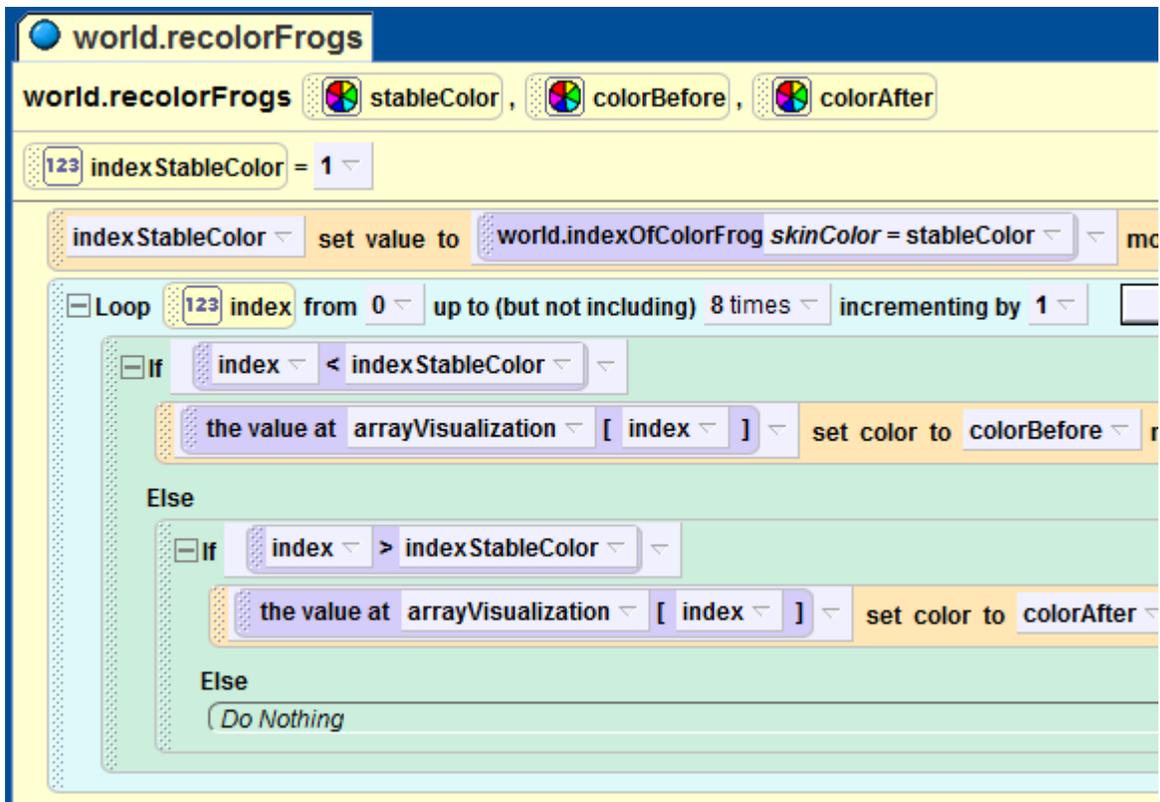
9. A) code is:



The image shows a Scratch code editor for a function named `world.indexOfColorFrog`. The function has one parameter, `skinColor`, represented by a color wheel icon. Below the parameter is the text "No variables". The code consists of a loop that iterates from index 0 to the size of `arrayVisualization`, incrementing by 1. Inside the loop, there is an if statement that checks if the color of the frog at the current index is equal to `skinColor`. If true, it returns the index. If false, it does nothing. After the loop, it returns -1.

```
123 world.indexOfColorFrog
world.indexOfColorFrog skinColor
No variables
Loop 123 index from 0 up to (but not including) arrayVisualization's size incrementing by 1
  If the value at arrayVisualization [ index ] .color == skinColor
    Return index
  Else
    Do Nothing
Return -1
```

B) Code is:



The image shows a Scratch code editor for a function named `world.recolorFrogs`. The function has three parameters: `stableColor`, `colorBefore`, and `colorAfter`, each represented by a color wheel icon. The code starts by setting a local variable `indexStableColor` to 1. Then, it enters a loop that runs 8 times, incrementing by 1. Inside the loop, there is an if statement that checks if the current index is less than `indexStableColor`. If true, it sets the color of the frog at that index to `colorBefore`. If false, there is another if statement that checks if the current index is greater than `indexStableColor`. If true, it sets the color of the frog at that index to `colorAfter`. If neither condition is met, it does nothing.

```
123 world.recolorFrogs
world.recolorFrogs stableColor, colorBefore, colorAfter
123 indexStableColor = 1
indexStableColor set value to world.indexOfColorFrog skinColor = stableColor
Loop 123 index from 0 up to (but not including) 8 times incrementing by 1
  If index < indexStableColor
    the value at arrayVisualization [ index ] set color to colorBefore
  Else
    If index > indexStableColor
      the value at arrayVisualization [ index ] set color to colorAfter
    Else
      Do Nothing
```

## Alternate Solution

The image shows a Scratch code editor window titled "world.recolorFrogs2". The code is as follows:

```
world.recolorFrogs2 stableColor, colorBefore, colorAfter
indexStableColorFrog = 1
indexStableColorFrog set value to world.indexOfColorFrog skinColor = stableColor more...
Loop 123 index from 0 up to (but not including) indexStableColorFrog times incrementing by 1 show simple version
  the value at arrayVisualization [ index ] set color to colorBefore more...
Loop 123 index from ( indexStableColorFrog + 1 ) up to (but not including) arrayVisualization 's size incrementing by 1
  the value at arrayVisualization [ index ] set color to colorAfter more...
```

The code implements an alternate solution for recoloring frogs. It starts by setting a variable `indexStableColorFrog` to 1. Then, it sets the value of `indexStableColorFrog` to the index of the frog whose skin color is `stableColor`. A loop then iterates from index 0 to `indexStableColorFrog` (exclusive), setting the color of each frog to `colorBefore`. A second loop iterates from `indexStableColorFrog + 1` to the end of the `arrayVisualization`, setting the color of each frog to `colorAfter`.