CompSci 4 - Test 2 – Nov. 14, 2006

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

"
Format of instructions:

Chicken properties and methods:
Given below are the chicken functions.

Array Visualization special methods and functions
1. (8 pts) Consider the following snapshot of an Alice world in which the cow is further than 6 meters away from the horse.

   A) List the events in this world.
   B) List the event handlers in this world.
   C) What happens in the world when the Play button is selected.

2. (4 pts) Consider the following array of animals shown below as a picture and as a property, and the code. When an object is printed, its name is printed. The first statement in the code below prints the word Chicken.

What values are printed out after the word Chicken?
3. (6 pts) Consider the following code that does not work as intended. This code should move `beachBall` a random distance up either 2 or 3 meters. Then move `beachBall` down the same amount to where it was originally.

List and explain any errors and how to fix them so the code works as intended.

4. (6 pts) Consider executing the following Alice code.

A. How many meters does the tortoise move?
   a. 1  b. 5  c. 10  d. 15  e. 50

B. How many meters does the whiteRabbit move?
   a. 1  b. 5  c. 10  d. 15  e. 50
5. (8 pts) Consider the following Alice world in which the pterodactyl is swooping down on a scorpion while a trex watches.

The pterodactyl is already pointing at the scorpion. Give the Alice code that does the following. Repeatedly move the pterodactyl 0.25 meters towards the scorpion while the scorpion waves his claw one time and doesn’t move (the scorpion has a method called “waveClaw” to wave his claw one time). Repeat moving the pterodactyl and waving the scorpion’s claw until the pterodactyl is < 0.5 meters from the scorpion. At this point assume the pterodactyl and the scorpion are touching and the pterodactyl should take off into the air with the scorpion.

6. (10 pts) Consider the following 3D text called score in which 5 is suppose to be added to the score whenever the kangaroo is clicked on and 2 is suppose to be added to the score whenever the frog is clicked on. It doesn’t work correctly.

To fix this a new method called incrementScoreNew with three parameters is created to replace incrementScore above. See how it is used in the Events window below. Write the code for incrementScoreNew below. Do not make any other modifications. Again, it should increment the score by 5 if the kangaroo is clicked on and by 2 if the frog is clicked on, and display the score. Nothing should happen if anything else is clicked on.
7. (20 pts) Assume there is a list called `frogs` containing six frogs of different colors as shown in the figure below.

![Frogs Image]

A. Write the function `HowManyOfThisColor` that has a parameter named `color` of type `color` and returns the number of frogs in the list of this type of color. For example, there are two blue frogs in the list above, so calling this function with `color` set to blue returns 2.

```world
world.HowManyOfThisColor(Blue)
```

B. The picture on the previous page also has a `balletBar` in the picture.

Write the method `MoveBehindBar` that has a parameter named `color` of type `color` that moves all the frogs in the list of that color type behind the balletBar. The frogs can be moved behind the balletBar by turning them around 180 degrees and then moving them forward (use `move`) the distance to the balletBar plus one meter. The picture above shows after the call to `MoveBehindBar` with `color` set to yellow, the four yellow frogs turned around and moved behind the balletBar. Complete the code below.

```world
world.MoveBehindBar(Yellow)
```
8. (20 pts) Consider the array of animals called `elements` shown below.

```
Array/Visualization's details

<table>
<thead>
<tr>
<th>properties</th>
<th>methods</th>
<th>functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>elements</code></td>
<td>boar, camel, bunny, cat, Chicken, flamingo</td>
<td></td>
</tr>
</tbody>
</table>
```

A) Write the function `World.firstShorterAnimal` that has a parameter called `someAnimal` and does the following. It returns the first animal in the array that is shorter than `someAnimal` (starting with the array position 0). For example, a call to `firstShorterAnimal` with `someAnimal` set to Boar returns Bunny, the first encountered animal that is shorter than Boar. If no animal in the array is shorter, then return `someAnimal`. The function has been started below. Don’t forget a return statement.

```
world.firstShorterAnimal
```

B) Write the function `World.indexOf` that does the following. It has an object parameter `someAnimal` and returns the index (or position of) `someAnimal` in the array. If `someAnimal` is not in the array, it returns –1.

For example, the call

```
World.indexOf someAnimal= Camel
```

returns 1 since camel is in slot 1 of the array.