Given below are the condition possibilities for an if statement

Below are the tiles at the bottom of a **procedure**

Below are the tiles at the bottom of a **function**
Given below are the panda procedures and panda Properties on the bottom right.
Given below are the panda functions.
If, loops, and changing an array element.

```plaintext
// Change a value in an array
this. creatures[1] = this.cheshireCat;
```
Events

```sql
declare procedure sceneActivated
do in order
  this myFirstMethod
```

```sql
declare procedure timeElapsed
do in order
  drop statement here
```

```sql
declare procedure keyPressed
  event isLetter event isDigit event getKey event isKey

do in order
  if event isKey $S$ is true then
    drop statement here
  else
    drop statement here
```

More Events

**mouseClicked**

```java
addMouseClickOnObjectListener setVisuals newVisual[] { this.bunny, this.panda, this.panda2, this.panda3 }
```

doen procedure

```java
do in order
if event getModelAtMouseLocation == this.panda is true then
    drop statement here
else
    drop statement here
```

**pointOfViewChanged**

```java
addPointOfViewChangeListener new SThing[] { this.bunny }
```

doen procedure

```java
do in order
    drop statement here
```

**collisionStarted**

```java
addCollisionStartListener new SThing[] { this.bunny }, new SThing[] { this.panda, this.panda2, this.panda3 }
```

doen procedure

```java
do in order
    drop statement here
```
1. (10 pts) Consider the following snapshot of an Alice world in which there are four objects standing from left to right on an invisible line: **marchHare**, **madHatter**, **bunny**, and **hare**. There are three units between the marchHare and madHatter, 1 unit between the madHatter and bunny, and 2 units between the bunny and the hare.

A) List the events in the world.

B) Explain everything that happens in the world when only the Run button is pressed.
C) After the world has played for 30 seconds, nothing should be moving. Explain how the user can interact with the world, and what happens in the world when the user interacts with it.
2. (12 pts) Consider the following array of bipeds called **friends**.

A) For the following code segment, explain what happens when this code executes. Be sure to make it clear who moves, when they move and where they move to, and if different creatures move, the order they move.
B) For the following code segment, explain what happens when this code executes with the array shown on the previous page. Be sure to make it clear who moves, when they move and where they move to, and if different creatures move, the order they move.

C) For the following code segment, explain what happens when this code executes with the array shown on the previous page. Note a hare has also been added to the world. Be sure to make it clear who moves, when they move and where they move to, and if different creatures move, the order they move.
3. Chickens (14 pts) - PART A Consider the following Scene procedure called **mysteryProc**. Assume **chickens** is an array that has only chickens in it.

A) How many times does the count loop execute if the chickenArray contains 4 chickens?

B) Explain what the procedure mysteryProc does.

C) If the “if indexChicken < this.chickenArray.length” IF is dissolved (that means the if is gone but the code inside of that IF is still there as the code in the loop), does the code execute the same result for when the IF is there? Explain.
PART B Consider the following Scene function called mystery. Assume chickens is an array that has only chickens in it.

A) What type of value does the function mystery return?

B) What is oneChicken: a property or a variable?

C) Explain what the function mystery does and what it returns.
4) (9 pts) Consider the following Alice world that shown below from left to right is a bunny, pig, tortoise and blackCat. The bunny is 3 units from the pig, the pig one unit from the tortoise, and the tortoise one unit from the blackCat. A mouseClicked event is also shown.

![Alice world diagram](image)

```
declare procedure mouseClicked
  event getModelAllMouseLocation
  event getScreenDistanceFrom
  event getMouseAllDistanceFrom

do in order
  if (event getModelAllMouseLocation == this.tortoise) is true then
    // Code
  else
    if (event getModelAllMouseLocation getDistanceTo this.tortoise <= 1.5) is true then
      // Code
    else
      // Code
  end if
end if
```

a. When this Alice world runs, what happens when the bunny is clicked on?

b. When this Alice world runs, what happens when the pig is clicked on?

c. When this Alice world runs, what happens when the blackCat is clicked on?

d. When this Alice world runs, what happens when the tortoise is clicked on, followed by the bunny clicked on?
5) (12 pts) Consider the following Alice world that has peacocks and penguins in it. The game has the peacocks and penguins move around randomly (that code is not shown). The user gets one point for clicking on a penguin and no points for clicking on a peacock. A score is shown that updates by one whenever a penguin is clicked on. Shown are the initial picture, some of the events, myFirstMethod and initializing and updating the score. Also shown is the array named <code>penguinArray</code> and a <code>textString</code> property named <code>gameState</code>.

Answer questions about this program on the next page.
A) Can one click on any penguin while the penguin is saying the instructions “Click on penguins to score” and receive a point? Explain.

B) After the penguin gives instructions, a user clicks on two penguins and receives 1 point for each. Then the user clicks on a peacock and receives one point. But the user is not suppose to receive any points for clicking on a peacock. This is a bug! Explain how to fix the code so clicking on a peacock does not update the score.

C) Suppose we want to modify updateScore so you can call it with different values for updating the score. For example, maybe you want to call it with 1 sometimes to add one point, 2 sometimes to add two points, 3 sometimes to add three points, etc. Explain how to modify updateScore so you can call it with any number and that number is added to the current score.
D) Suppose we change the game so that the user can click on each penguin only three times. The first time a user clicks on a penguin they receive 3 points and the penguin fades to .70 percent opacity. The second time a user clicks on the same penguin they receive 2 points and that penguin fades to .40 percent opacity. The third time a user clicks on the same penguin, they receive 1 point and that penguin becomes invisible. Modify the addMouseClickOnObjectListener to make these change and give the new code. (You may use the change you made in part C).
6) (8 pts) Consider the following Alice world with a hare, chicken, pig and phoenix, and the addCollisionStartListener and the AddDefaultModelManipulation.

Answer questions about this code/world on the next page.
A. When the Alice world runs, what happens when the user clicks on the chicken and drags the chicken into the pig?

B. When the Alice world runs, what happens when the user clicks on the pig and drags the pig into the hare?

C. When the Alice world runs, what happens when the user clicks on the pig and drags the pig into the phoenix?

D. When the Alice world runs, what happens when the user clicks on the chicken and drags the chicken into the hare?
7) (4 pts) Consider the following Alice program where all the objects shown are in the array named **friends**. The function **widest** is supposed to return the widest object in the array **friends**. When **myFirstMethod** runs the widest method returns the pig, who is NOT the widest object in the array. The elk is the widest.

```
unmanaged
  SJointedModel[] friends <- new SJointedModel[] { this.pig, this.desertTortoise, this.bear, this.elk, this.yari }
```

```
declare procedure myFirstMethod
  do in order
    (this) Widest move UP 1.0 add detail
  (this) Widest move DOWN 1.0 add detail
```

```
declare SJointedModel function Widest
  do in order
    SJointedModel WidestSoFar <- (this) friends [0]
    for each (SJointedModel) someone in (this) friends
      if (someone) getWidth > (widestSoFar) getWidth is true then
        (widestSoFar) <- (someone)
      else
        (widestSoFar) <- (someone)
    return (widestSoFar)
```

Explain how to fix the widest function so that it always returns the widest object in the array and show the code (you can modify the code above). Your function should work even if the elements in the array are changed.
8) (12 pts) Consider an Alice world with one peacock and an array of penguins named penguinArray. The penguins in the array may have different visibilities, either completely visible, partially visible or invisible.

A. Write the Scene function `NumberWithVisibilityInRange` that has two parameters. The first one is a `DecimalNumber` named `minVisibility`, and the second one is a `DecimalNumber` named `maxVisibility`. This function should return the number of penguins whose visibility is in the range from `minVisibility` to `maxVisibility` inclusive.

```
declare function NumberWithVisibilityInRange
with parameters: DecimalNumber = minVisibility, DecimalNumber = maxVisibility
```
B. Write the Scene procedure called ComparePenguinsWithDifferentVisibility that has been started below for you. This procedure figures out which there are more of in the array: those completely visible, those partially visible or those invisible. Then the peacock says “There are more XXX penguins” where XXX is either “visible”, “partially visible”, or “invisible.” For example, for the picture on the previous page, the peacock would say “There are more partially visible penguins” since there are 4 partially visible penguins which is a larger number than the number of visible (2) or invisible (0) penguins. If there is a tie, (say 2 of each type of visibility, all tied) you can say either one that is the tie, “There are more visible penguins.” You don’t have to handle for when there is a tie.
(extra page 1, must turn in)
(extra page 2, must turn in)