

# CPS 94 Test 1 Version 1 Fall 2018

Given below are the condition possibilities for an if statement

The image shows the 'if' statement menu in Scratch, which is divided into two main sections: 'true' (current value) and 'false'. The 'true' section includes options for 'true', 'false', 'nextRandomBoolean', 'NOT true', 'NOT ???', 'BOTH true AND ???', 'EITHER true OR ???', 'BOTH ??? AND ???', and 'EITHER ??? OR ???'. The 'false' section includes options for 'Relational (DecimalNumber) { ==, !=, <, <=, >=, > }', 'Relational (WholeNumber) { ==, !=, <, <=, >=, > }', 'Relational (SThing) { ==, != }', and 'TextString Comparison'. To the right of the menu are three panels of condition options. The first panel contains: '??? <= ???', '??? < ???', '??? >= ???', '??? > ???', '??? == ???', and '??? != ???'. The second panel contains: '??? == ???' and '??? != ???'. The third panel contains: '??? contentEquals ???', '??? equalsIgnoreCase ???', '??? startsWith ???', '??? endsWith ???', and '??? contains ???'. Arrows point from the 'if' menu to these three panels.


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

The image shows the tiles at the bottom of a procedure block in Scratch. The tiles are: 'do in order', 'count \_', 'while \_', 'for each in \_', 'if \_', 'do together', 'each in \_ together', 'variable...', 'assign', and '//comment'.

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

The image shows the tiles at the bottom of a function block in Scratch. The tiles are: 'do in order', 'count \_', 'while \_', 'for each in \_', 'if \_', 'do together', 'each in \_ together', 'variable...', 'assign', '//comment', and 'return \_'. A blue scribble is drawn over the tiles.

Given below are the panda procedures and panda Properties on the bottom right.

 **this.panda**

Procedures

Functions

group by category ▼

Panda's Editable Procedures (3)

edit

this.panda

standingPose

edit

this.panda

sleepingPose

edit

this.panda

crawlingPose

Biped's Editable Procedures (0)

say, think

this.panda

say

text: ???

this.panda

think

text: ???

position

this.panda

move

direction: ???

amount: ???

this.panda

moveToward

target: ???

amount: ???

this.panda

moveAwayFrom

target: ???

amount: ???

this.panda

moveTo

target: ???

this.panda

place

spatialRelation: ???

target: ???

orientation

this.panda

turn

direction: ???

amount: ???

this.panda

roll

direction: ???

amount: ???

this.panda

turnToFace

target: ???

this.panda

orientTo

target: ???

this.panda

orientToUpright

this.panda

pointAt

target: ???

position & orientation

this.panda

moveAndOrientTo

target: ???

size

this.panda

setWidth

width: ???

this.panda

setHeight

height: ???

this.panda

setDepth

depth: ???

this.panda

resize

factor: ???

this.panda

resizeWidth

factor: ???

this.panda

resizeHeight

factor: ???

this.panda

resizeDepth

factor: ???

appearance

this.panda

setPaint

paint: ???

this.panda

setOpacity

opacity: ???

vehicle

this.panda

setVehicle

vehicle: ???

audio

this.panda

playAudio

audioSource: ???

timing

this.panda


delay

duration: ???

other


this.panda

straightenOutJoints

 **this.panda**

one shots ▼

▼

 **this.panda's Properties**

Panda

panda

←

new Panda

Paint = 

WHITE

Opacity = 

1.0

Vehicle = 

this

Position = ( x: -1.00 , y: 1.73 , z: -0.10 )

Width: 

0.75

Size = Height: 

1.15

Depth: 

0.53

Reset

Show Joints: ☐

Given below are the panda functions.

**this.panda**

**Procedures Functions**

group by category ▼

Panda's Editable Functions (3)

- ☐ edit **this.panda** getLeftEar
- ☐ edit **this.panda** getRightEar
- ☐ edit **this.panda** creatureAbove friend1: ???, friend2: ???

Biped's Editable Functions (0)

appearance

- this.panda** getPaint
- this.panda** getOpacity

size

- this.panda** getWidth
- this.panda** getHeight
- this.panda** getDepth

prompt user

- this.panda** getBooleanFromUser message: ???
- this.panda** getStringFromUser message: ???
- this.panda** getDoubleFromUser message: ???
- this.panda** getIntegerFromUser message: ???

other

- this.panda** getDistanceAbove other: ???
- this.panda** getDistanceBehind other: ???
- this.panda** getDistanceBelow other: ???
- this.panda** getDistanceInFrontOf other: ???
- this.panda** getDistanceTo other: ???
- this.panda** getDistanceToTheLeftOf other: ???
- this.panda** getDistanceToTheRightOf other: ???
- this.panda** getVantagePoint entity: ???
- this.panda** getVehicle
- this.panda** isAbove other: ???
- this.panda** isBehind other: ???
- this.panda** isBelow other: ???
- this.panda** isCollidingWith other: ???
- this.panda** isFacing other: ???
- this.panda** isInFrontOf other: ???
- this.panda** isToTheLeftOf other: ???
- this.panda** isToTheRightOf other: ???
- this.panda** toString

**joints**

- this.panda** getHead
- this.panda** getLeftAnkle
- this.panda** getLeftClavicle
- this.panda** getLeftElbow
- this.panda** getLeftEye
- this.panda** getLeftEyelid
- this.panda** getLeftFoot
- this.panda** getLeftHand
- this.panda** getLeftHip
- this.panda** getLeftIndexFinger
- this.panda** getLeftIndexFingerKnuckle
- this.panda** getLeftKnee
- this.panda** getLeftMiddleFinger
- this.panda** getLeftMiddleFingerKnuckle
- this.panda** getLeftPinkyFinger
- this.panda** getLeftPinkyFingerKnuckle
- this.panda** getLeftShoulder
- this.panda** getLeftThumb
- this.panda** getLeftThumbKnuckle
- this.panda** getLeftWrist
- this.panda** getMouth
- this.panda** getNeck
- this.panda** getPelvis
- this.panda** getRightAnkle
- this.panda** getRightClavicle
- this.panda** getRightElbow
- this.panda** getRightEye
- this.panda** getRightEyelid
- this.panda** getRightFoot
- this.panda** getRightHand
- this.panda** getRightHip
- this.panda** getRightIndexFinger
- this.panda** getRightIndexFingerKnuckle
- this.panda** getRightKnee
- this.panda** getRightMiddleFinger
- this.panda** getRightMiddleFingerKnuckle
- this.panda** getRightPinkyFinger
- this.panda** getRightPinkyFingerKnuckle
- this.panda** getRightShoulder
- this.panda** getRightThumb
- this.panda** getRightThumbKnuckle
- this.panda** getRightWrist
- this.panda** getSpineBase
- this.panda** getSpineMiddle
- this.panda** getSpineUpper

1. (3 pts) Which one of the following HTML tags does not result in bold letters?

- A) <b>
- B) <em>
- C) <h2>
- D) <strong>

2. (3 pts) Consider the following list that has been created with HTML.

- UNC
  - NCSU
1. WFU
  2. **Duke**

Which one of the following is the HTML that generated this list?

- |   |   |
|---|---|
| A) <ul><br><li> UNC </li><br><li> NCSU </li><br><ol><br><li> WFU </li><br><li> Duke </li><br></ol><br></ul> | B) <ol><br><li> UNC </li><br><li> NCSU </li><br><ul><br><li> WFU </li><br><li> Duke </li><br></ul><br></ol> |
| C) <ul><br><ol><br><li> UNC </li><br><li> NCSU </li><br></ol><br><li> WFU </li><br><li> Duke </li><br></ul> | D) <ol><br><ul><br><li> UNC </li><br><li> NCSU </li><br></ul><br><li> WFU </li><br><li> Duke </li><br></ol> |

3. (3 pts) Consider the following HTML code. Draw what it would look like on a web page.

```
<table border=1>
<tr>
<td> A </td></tr><tr><td> B </td></tr>
<tr><td> C </td> </tr> <tr> <td> D </td></tr>
<tr><td> E </td> </tr> <tr><td> F </td>
</tr>
</table>
```

4. (3 pts) Consider the following CSS for table.

```
table {
  color: red;
}

table td {
  color: blue;
}
```

Explain how these two will affect how a table is displayed.

5. (4 pts) Consider the following that appears in a .css file.

```
#largeFont {  
    font-size: 64px;  
}
```

- a) Is this an ID or a Class?
- b) Consider the .html file that refers to this .css file. Give the .html that shows how one applies the CSS above to the word “football” in the following line in the .html file.

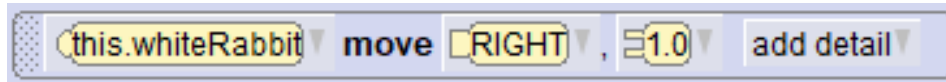
Where are the football fans?

6. (4 pts) A six-digit hexadecimal number represents a color. A color can also be represented as a three-tuple of numbers representing the red, green and blue components (or RGB) of the number. **You must show your work for full credit!**

A. Convert the hexadecimal number 061B23 into RGB: (       ,       ,       )

B. Convert the color in RGB (50, 13, 27) to the corresponding six-digit hexadecimal number:

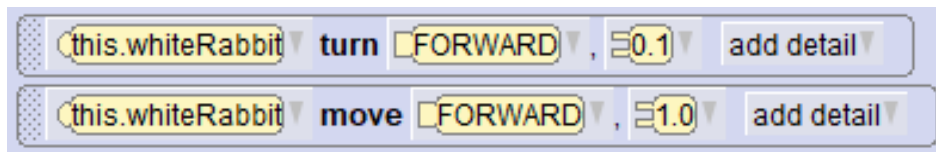
7. (3 pts) Consider the following Alice code and the starting position of the whiteRabbit and panda that is shown with **Start** in the figure on the left below.



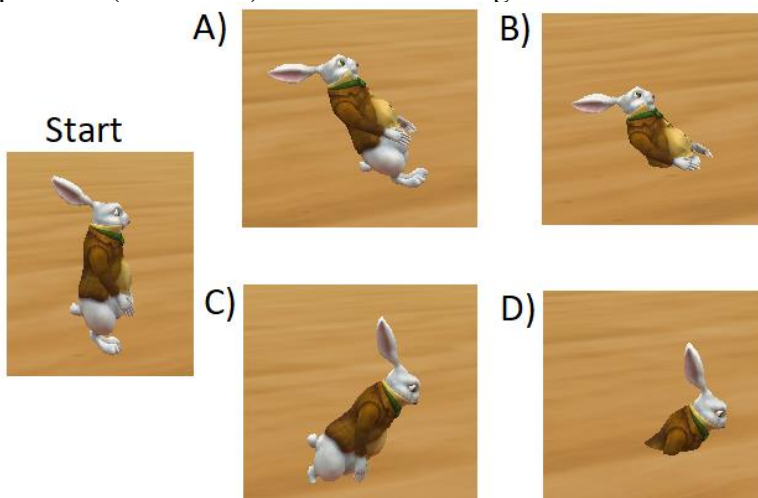
Which one of the following shows the result after starting in the start position (on the left) and then executing this Alice code?



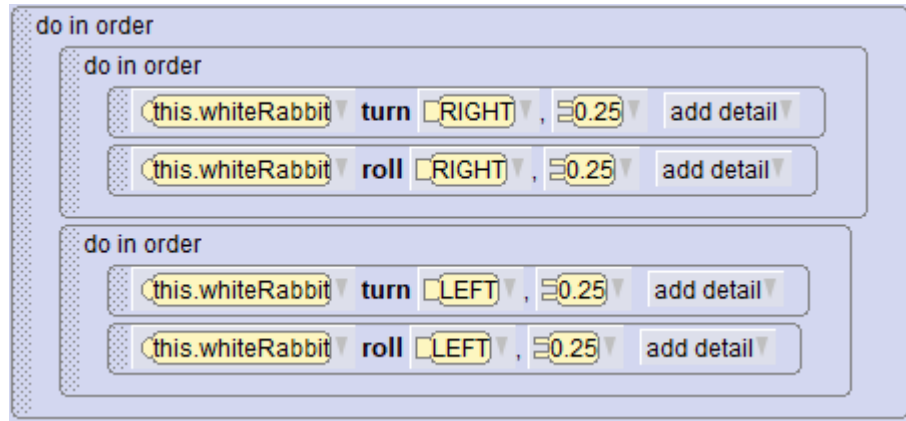
8. (3 pts) Consider the following Alice code and the starting position of the whiteRabbit and panda that is shown with **Start** in the figure on the left below.



Which one of the following shows what the whiteRabbit looks like starting in the start position (on the left) and then executing this Alice code?



9. (3 pts) Consider the following Alice code and the starting position of the whiteRabbit that is shown with **Start** in the figure on the left below.



Which one of the following shows what the whiteRabbit looks like starting in the start position (on the left) and then executing this Alice code?

Start



A)



B)



C)



D)



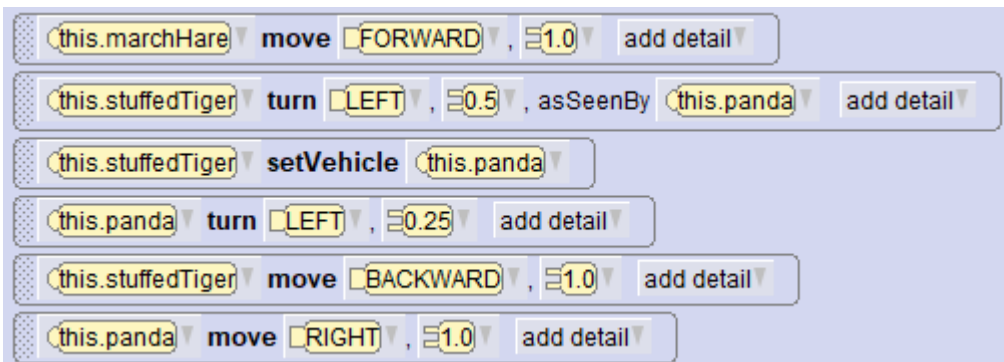


10. (14 pts) Consider the following Alice code in which the lines are numbered.

```
1  this.panda setPaint this.panda something item: 2 add detail
2  if this.bunny isBehind this.pig add detail is true then
3    this whichOne name: this.bunny word: superDog turn RIGHT 2.0 add detail
4  else
5    this.pig turn RIGHT 0.5 add detail
6  this tricky who: this.pig way: BACKWARD
```

- A) In line 1, is “something” a function or a procedure?
- B) In line 2, what TYPE is “this.bunny isBehind this.pig”?
- C) In line 3, what is the name of the function and what TYPE does it return?
- D) What must be true so that line 5 executes?
- E) In line 6, list the word(s) that are arguments.
- F) In line 6, list the word(s) that are parameters.
- G) In line 6, is “tricky” a function or a procedure?

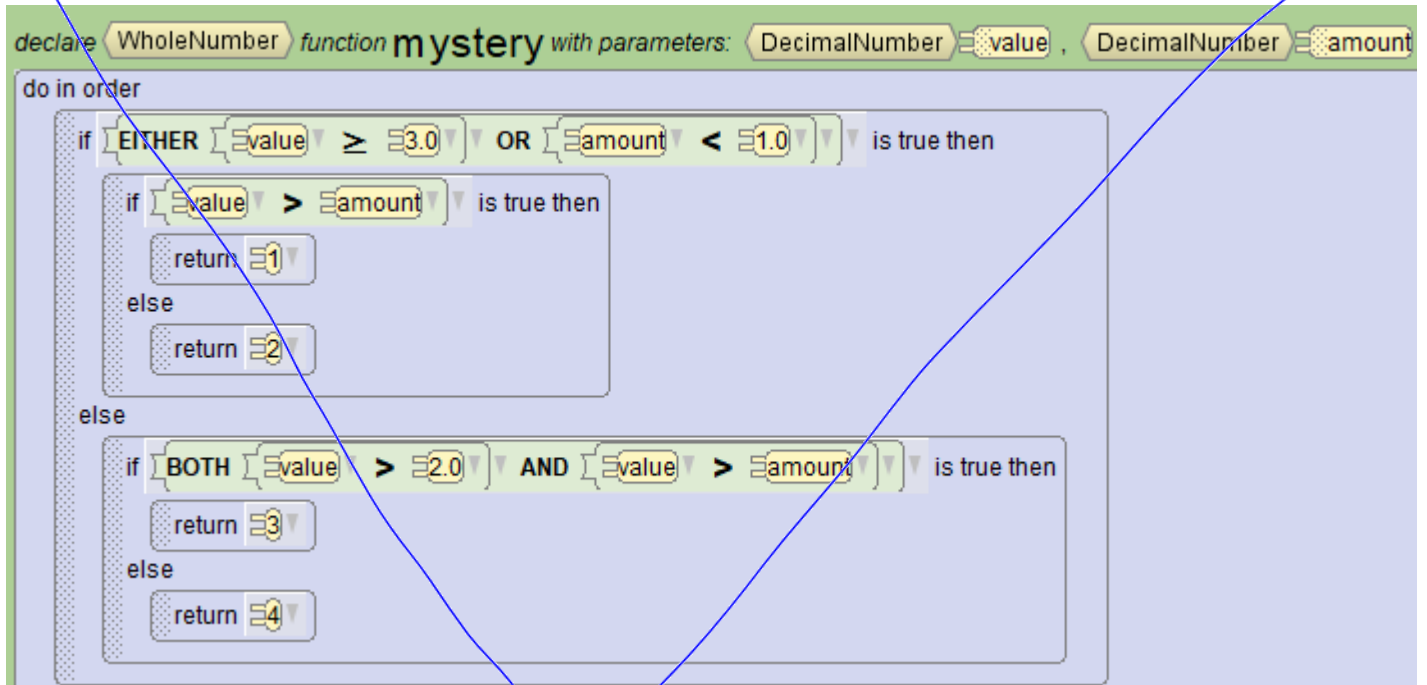
11. (6 pts) Consider the following world that has the three objects: stuffedTiger, marchHare and panda (shown below from left to right) and the given code. The world has been setup as shown below. The stuffedTiger is **exactly 1.0 meter** from the marchHare, and the panda is **exactly 1.0 meter** from the marchHare.



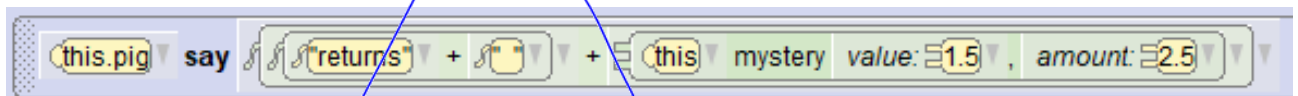
The diagram below is looking from above over the scene. The stuffedTiger is represented by the S, the marchHare is represented by the M, and the panda is represented by the P. The animals are facing the bottom of the page. Using the diagram below, draw the path of stuffedTiger and marchHare as a solid line and the path of panda as a dashed line.

S      M      P

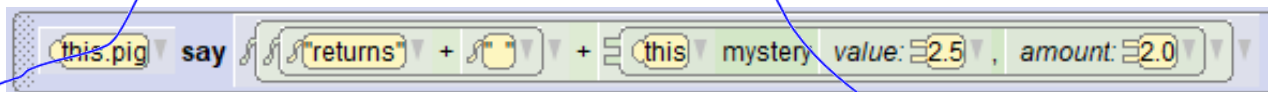
12. (4 pts) Consider the following `mystery` function.



A) What does `pig` say when the following line of code is run?



B) What does `pig` say when the following line of code is run?



13. (10 pts) Consider the following Alice world that has three objects: `cow`, `cedarPole` and `camel`.



The world starts as shown in the figure above with the cow and camel facing each other and also facing the cedarPole. Assume there is a cameraMarker called `cameraStart` that is the position the camera is currently shown in. When you move the animals you do not need to move their legs, just move them. **Write code** to do the following in this order.

- Move the camera so it is above looking down on the animals.
- Have the cow and camel turn twice around the cedar pole at the same time, going the same direction and moving with their head first (not sideways).
- Move the camera back to the camera starting view.

```
declare procedure myFirstMethod
do in order
```

14. (9 pts) Complete the following Alice **procedure** called `GetInLine` whose header is shown on the next page. This procedure is called several times to build a line of creatures lined up behind Alice. In particular, this Alice procedure has three parameters, an `SJointedModel` named “creature,” a decimal number named “howFar,” and an `SJointedModel` named “lastCreature”. Assume the Alice world starts with Alice facing three creatures who are lined up, all facing the camera as shown below. This procedure has *creature* line up behind Alice, more specifically placed “howFar” behind the “lastCreature” who is the last one in line. Once the creature is in place, code is set so that if Alice moves, the creature will also move.



Here are three calls to this procedure followed by Alice moving forward with everyone following.

```

declare procedure myFirstMethod
do in order
  this.alice getInLine creature: this.tortoise , howFar: 0.5 , lastCreature: this.alice
  this.alice getInLine creature: this.panda , howFar: 0.5 , lastCreature: this.tortoise
  this.alice getInLine creature: this.bunny , howFar: 1.0 , lastCreature: this.panda
  this.alice move FORWARD , 10.0 add detail

```

The result of this code is that the tortoise moves to 0.5 behind Alice, then the panda moves to 0.5 behind the tortoise, who is last in line, then the bunny moves to 1.0 behind the panda, who is last in line. Then in the last line of code Alice moves forward and everyone in line follows her.

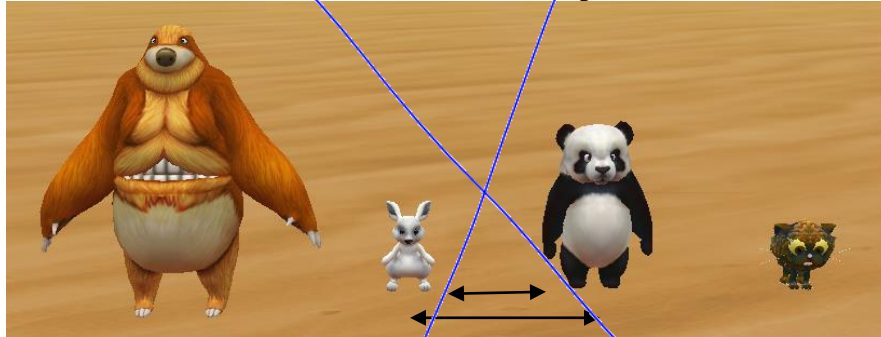


Complete the code below.

```
declare procedure getInLine with parameters: SJointedModel creature ,  
DecimalNumber howFar , SJointedModel lastCreature
```

15. (10 pts) Assume there are four objects in an Alice world, a mapinguari, a bunny, a panda, and a blackCat (shown in that order left to right below) all facing the camera. Complete the following **bunny function** called `fitBetween` that has two STurnable parameters, one named `friend`, and one named `object`. This function returns True if `object` can “fit between”. or stand between bunny and friend, while the object is also facing the camera, and returns False otherwise.

Hint: Note that the “distanceTo” function measures the distance between two objects from the center of their bodies, the bottom double arrow below. You want to measure the distance between them, the top double arrow below.

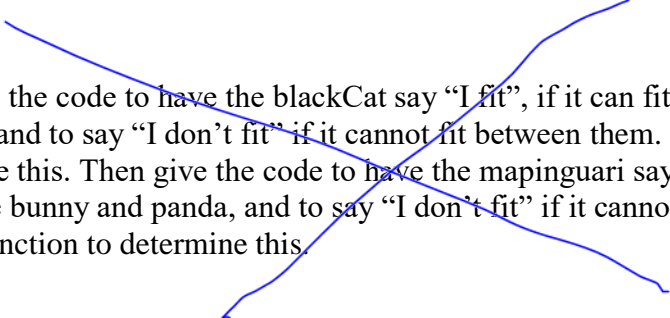


Complete the function on the next page.

A) (7 pts) Complete the **bunny** function below.

declare Boolean function fitBetween with parameters: SJointedModel friend, SJointedModel object





B) (3 pts) Give the code to have the blackCat say “I fit”, if it can fit between the bunny and panda, and to say “I don’t fit” if it cannot fit between them. Use your function to determine this. Then give the code to have the mapinguari say “I fit”, if it can fit between the bunny and panda, and to say “I don’t fit” if it cannot fit between them. Use your function to determine this.