Algebra 1 – WH names \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Unit 6: Class directions date 10/16/2014 to 10/17/2014 pd \_\_\_\_

1. Warmup: Evaluate each of the following code expressions.

|  |  |  |
| --- | --- | --- |
| # | Expression | Evaluate  (get an answer) |
| 1. | ( + 6 9) |  |
| 2. | ( - 5 7) |  |
| 3. | ( = 3 8 ) |  |
| 4. | ( = (+ 2 3) (\* 1 5)) |  |
| 5. | ( string=? “bootstrap” “programs”) |  |
| 6. | ( >= 3 4) |  |
| 7. | ( = “blue” 9) |  |

1. Practice: Convert each math expression to a circle of evaluation and to code.

*(grab one laptop per group and log on while doing this practice)*

|  |  |  |  |
| --- | --- | --- | --- |
| # | Math Expression | Circle of evaluation | Code |
| 1. | 10 = 16.1 |  |  |
| 2. | -13 > 5 |  |  |
| 3. | 40 – 1 < 90 \* 2 |  |  |
| 4. | 0 = -5 + 10 |  |  |

1. Write contracts and examples for the four following functions in the back of your workbook:

< , > , = , string=?

1. Onscreen?

Open the "Cage" program (it’s on the Wikispace) and click "Run". The window that pops up contains a butterfly, which you can move around using the arrow keys on your keyboard. This butterfly is named Sam, and the window is Sam’s yard. Much like the videogame screen, this yard is 640 pixels wide by 480 pixels tall! As you can see, Sam is free to walk around the yard, and the coordinates at the top of the screen update to tell you his location.

Sam’s mother tells him that it’s okay for him to step outside of the yard, but only by a short distance! Specifically, she wants to make sure that she can always see at least a little piece of him. **Sam is safe *as long as some piece of him is onscreen*.** That means he can go a little past zero on the left-hand side, or a little past 640 on the right.

How far *can* he safely go on the left? \_\_\_\_\_\_\_\_\_\_ On the right? \_\_\_\_\_\_\_\_\_\_\_



1. Turn to [Page 19](http://www.bootstrapworld.org/materials/Fall2014/courses/bs1/resources/workbook/StudentWorkbook.pdf) in your workbook and read through the top half of the page. Using the arrow keys, see how far Sam can go to the left *while still staying onscreen*. You’ll find that Sam is onscreen as long as his x-coordinate is *greater than* -50, which is exactly the code that’s filled in for the first problem on the page.
   * + Fill in the code for the second problem.
     + Convert both of those expressions into Circles of Evaluation, using the empty Circles at the bottom of the page.



1. Unfortunately, right now there is nothing to keep Sam from leaving his yard! Looking at the code for this program, we find three function definitions:
   * 1. safe-left?safe-left?, which takes in Sam’s x-coordinate and checks to see if Sam is safe on the lefthand side of the screen. Right now, this function always returns truetrue.
     2. safe-right?safe-right?, which takes in Sam’s x-coordinate and checks to see if Sam is safe on the righthand side of the screen. Right now, this function always returns truetrue.
     3. onscreen?onscreen?, which takes in Sam’s x-coordinate and checks to see if at least tiny piece of Sam is on the screen. Right now, this function *asks* safe-left?*safe-left? if the given x-coordinate is safe.*

Right now, all three functions return truetrue no matter what Sam’s x-coordinate is!

Discuss what you just read with your partner. Do you have any questions?

1. Right now safe-left?safe-left? is ignoring Sam’s x-coordinate, and blindly returning truetrue no matter where he is. What should the function be doing instead?

Turn to [Page 20](http://www.bootstrapworld.org/materials/Fall2014/courses/bs1/resources/workbook/StudentWorkbook.pdf) in your workbook. Use the Design Recipe to fix this function, so that safe-left?safe-left? takes in Sam’s x-coordinate and returns truetrue or falsefalse depending on whether it is *greater than -50*.

Once you have typed in the EXAMPLEs and definition for safe-left?, click "Run" and see if Sam is prevented from flying off the lefthand side of the screen. Be sure to test your function in the Interactions window as well, for example:

(safe-left? 50)

(safe-left? -100)

(onscreen? 50)

(onscreen? -100)



1. Turn to [Page 21](http://www.bootstrapworld.org/materials/Fall2014/courses/bs1/resources/workbook/StudentWorkbook.pdf) in your workbook. Use the Design Recipe to fix this function, so that safe-right?safe-right? takes in Sam’s x-coordinate and returns truetrue or falsefalse depending on whether it is *less than 690*.

Once you have typed in the EXAMPLEs and definition for safe-right?safe-right?, click "Run" and see if Sam is prevented from flying off the righthand side of the screen. You may be surprised at what happens! Test your function in the Interactions window:

(safe-right? 50)

(safe-right? 810)

(onscreen? 50)

(onscreen? 810)

Is safe-right?safe-right? working the way you expected? YES / NO

Is onscreen?onscreen? working as you expected? YES / NO

The onscreen?onscreen? function is only talking to safe-left?safe-left?, which means that Sam is only protected on the lefthand side of the screen. Change the code for onscreen?onscreen? so that it calls safe-right?safe-right? instead. Now Sam will be protected on the righthand side of the screen, but not the left.

What should onscreen?onscreen? be doing? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1.  You’ve already learned many functions that allow you to create Booleans. The functions << and >>, for example, will each return a boolean when applied to the appropriate input. What happens when you want to *combine* these tests, to see if more than one thing is true?

Consider the following statements - are they true or false?:

|  |  |
| --- | --- |
|  | True or false? |
| Sugar is sweet |  |
| Ice is hot |  |
| Sugar is sweet **and** Ice is cold |  |
| Sugar is sweet **and** Ice is hot |  |
| Sugar is sweet **or** Ice is hot |  |

Fill in the blanks of the contracts and purpose statements:

; and: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 🡪 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; Returns true if \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ inputs are true

; or: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 🡪 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; Returns true if \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ of the inputs are true

1. Write contracts and examples for the two following functions in the back of your workbook:

and , or

1. Complete page 22 in your workbook.



1. Onscreen? 2.0

Goal: Combine both tests to keep Sam onscreen.

Turn to Page 23 in your workbook and write the completed onscreen?onscreen? function, so that it tests both safe-left?safe-left? and safe-right?safe-right?. Type in the examples and definition when you’re done, click "Run", and see if Sam is protected on both sides!

Share your work with your teacher.



1. Open your game file, and scroll down past the update-update- functions until you see the definitions for safe-left?safe-left?, safe-right?safe-right? and onscreen?onscreen?. Right now, they all return true. Since your gamescreen has the same dimensions as Sam’s yard (640x480), you can use the same code for each of these three functions! Make sure you include EXAMPLEs, so that your game code can test each function separately.