

# 2015 HOPSCOTCH HOUR OF CODE



**Teacher-led Lesson Plan**

## TIME

45-60 minutes (+15 minutes of optional, free code time)

## BIG IDEA

Coding is a superpower! If you know how to use individual blocks like conditionals and variables, you can put them together in powerful ways to make what you want. Little blocks build big programs!

## SKILL FOCUS

- Planning and Execution
- Planning and carrying out investigations (NGSS Practice 3)
- Analyzing and interpreting data (NGSS Practice 4)
- Reason abstractly and quantitatively (CCSS.MATH.PRACTICE.MP2)
- Look for and express regularity in repeated reasoning (CCSS.MATH.PRACTICE.MP8)

## KEY VOCABULARY

**Sequence:** The order in which instructions are given to the computer

**Event:** When something happens

**Rule:** Instructions that tell your computer what to do (the command) and when to do it (the event)

**Value/Variable:** A number that can change

**Conditional:** statement of the form "IF (something is true) THEN (do an action)"

## TRANSFER GOALS

1. Students will understand that a conditional checks if something is true and can identify the use of conditionals in their lives.
2. Students will understand that a value or variable is a holder for a number and can identify the use of values in their lives.
3. Students will become comfortable with keeping track of variables and doing operations.
4. Students will make a plan and see it through until the end.
5. Students will recognize patterns in their code and apply them to new situations.

## MATERIALS

- 1 iPad per student, or 1 iPad per 2 students, for pair programming
- Complete project available:  
<http://hop.sc/1L5sH2Q>

Hi!

We're *really* excited that, this Hour of Code, you're programming with your students—both for them and for you. Kids have remarkable imaginations, and creating computer programs is an amazing way for them to express themselves. We've seen kids create astonishing things using our simple but powerful tool.

**Anyone, regardless of their experience with programming, can teach this lesson.** Just as Hopscotch was built on the principle that anyone can become a great programmer, this lesson is designed on the premise that anyone can become a great programming teacher.

In this lesson, students will create a if-you-chose-mostly-A's type quiz. In our example, it will help them answer an age-old question: "Which emoji are you?" With a little adjustment, it could be made into a factual quiz that checks the answers for correctness; for example, you could adapt it for an english class to test mastery of literary techniques or in biology to assess knowledge of cell functionals. This lesson introduces some of the most important computing concepts: **sequencing, events, loops, values, and conditional statements**. You can find a glossary of terms at the end of the lesson (page 16).

**This HOC activity is broken into smaller tasks**, each of which begins with a discussion of the programming task at hand and the core concepts evoked in completing it. You can use this context to lead conversations with your students or, if they have prior programming experience and a good grasp of these concepts, feel free to skip.

After a discussion of what needs to be built and, if desired, how it might be coded, students can start coding. Depending on how many iPads you have, you can have students work independently or in pairs. At Hopscotch, we do a lot of pair programming (where two engineers share one computer) because it helps us write smarter, less-buggy code. We recommend trying it! All students should get into the habit of testing their code frequently by running (playing) it. It is much easier to find and solve mistakes when you're constantly testing.

We demonstrate how each task can be implemented in Hopscotch with screenshots of sample code. The code we suggest usually is only one way to build the needed feature; there are often other ways that students can accomplish their goals.

Have fun and we can't wait to see what your students build. Share their projects on social media and tag with #madeonhopscotch and @hopscotch on Twitter and @gethopscotch on Instagram :-)

Yours,  
Jocelyn Leavitt  
Co-founder and CEO, Hopscotch

# ACTIVITY GUIDE

## 0. Discussion (5 minutes)

The first and most important lesson of computer science is that computers do what they are told, and only what they are told, in the order they are told to do it.

If you fully understand this concept and begin to think of everyday processes (making a sandwich, getting to school) as a set of instructions, you will begin to think like a programmer without trying very hard! A programmer is a person who codes, or writes computer programs. A program is a set of instructions a computer can understand. We refer to these instructions as a **sequence**. This term also refers to the idea that computers must follow the instructions in the order, or “sequence” in which they’re given.

Ask your students to name some programs they use. Consider all their games and apps, but also the software a DJ uses to mix tracks, the database your doctor uses to monitor your health, and the video games you play after school. All are programs and all were created by programmers.




How many times a day do you interact with computers? Are there computers in surprising places? How about a car? How about a phone? If you can control these computers and write programs for them, you can make things that millions of people use every day!

## 1. Make a plan (5 minutes)

The first step is to design the questions and answers for your quiz. Pass out copies of the blank matrix below, or put it on the projector and have students create their own on paper. Don't worry too much about making them the best questions ever; this is just your first quiz!

	Question 1	Question 2	Question 3
Result A	Answer 1A	Answer 2A	Answer 3A
Result B	Answer 1B	Answer 2B	Answer 3B
Result C	Answer 1C	Answer 2C	Answer 3C

Some ideas include: What job should you have? What fictional character are you most like? What book should you read next? If time is limited, or if a student has a hard time coming up with an idea, use the sample matrix below:

	What are you best at?	Where do you like to hang out?	Would you move to Mars if you could?
	Playing music	Library	No.
	Playing sports	Park	Yes!
	Telling jokes	Pool	Depends, do they have ice cream?

# ACTIVITY GUIDE

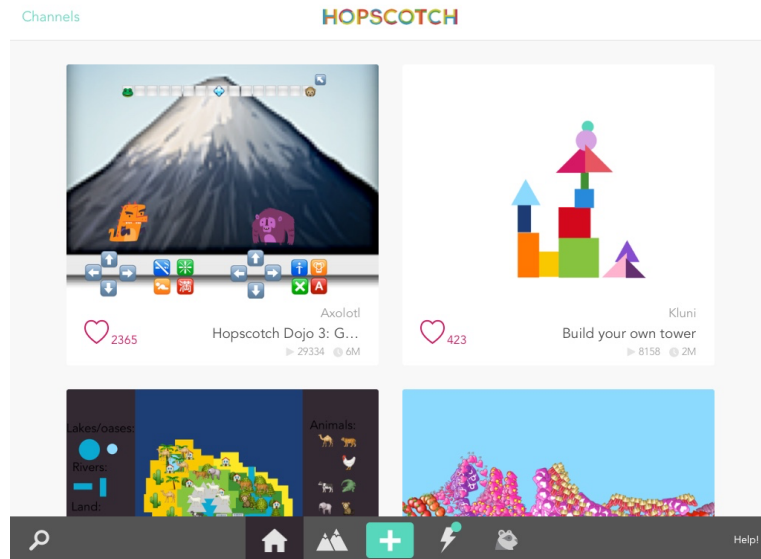
## 2. Using Hopscotch (5 minutes)

Download and get your students acquainted with Hopscotch. (<http://hop.sc/gethopscotch>)

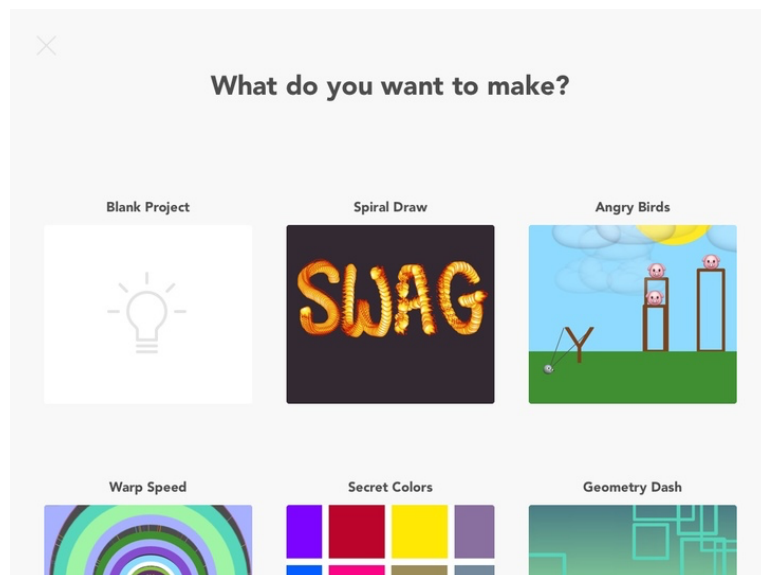
### 2.1 Finding the Hopscotch app on your iPad

### 2.2 Signing into your account (students may need to create accounts)

### 2.3 Making a new project: Tap on the green highlighted + on the bottom of the screen



### 2.4 Choose Blank Project



## 3. Set up (5 minutes)

The point of the quiz is to tell the player which emoji most represents them. The player will navigate through the quiz by tapping on questions and answers that appear as the player advances.

One of the most important lessons of this activity is learning that the programmer must not only put together all the components of a game (buttons, background, character), but also

# ACTIVITY GUIDE

explicitly tell the computer how they should work. For this, we need to create a **rule**, or code that tells the computer what to do and when to do it. A rule has two components: **an event** and **commands (or action)**.

**An event** is a trigger that the computer recognizes and causes it to do some action. In Hopscotch, all events start with the word “When” and are the first thing you choose when you write a rule. Think of it as completing a “WHEN....., THEN.....” sentence.

Events are deeply important for computer engineers because they tell the computer when it should do something. When you touch the phone icon on your home screen, then your phone brings up the interface to make calls. When an Angry Bird hits a block, then the block falls down.

Discuss some events (triggers) that happen in the classroom. Identify the trigger and resulting action: When I raise my hand (trigger), then stop talking (action), when the bell rings (trigger), then put down your pencil and turn in your test (action).

After you discuss, students can begin assembling their quiz by creating the necessary text objects.

## 3.1 Add text objects for the title, question, three answers (5 objects)



Tap on the grey “+” symbol in the upper right corner of the screen to get your objects. For each of your 5 objects, choose a text object, and type in the appropriate text.

## 4. Introduction to values (10 minutes)

Before coding, discuss how the quiz will work and the tools we will use to build it. The quiz has three main functions: First, it keeps track of what question you’re on and advances the question number when the player chooses an answer. Second, it keeps track of how many of each answer you’ve chosen. Third, it compares the answers at the end to give the player a result. We will use **values** to keep track of all this information.

**Values**, also known as **variables**, hold pieces of information. Values can be set, changed, or checked. They can be used inside **events** and other blocks, and can stand in any place you

# ACTIVITY GUIDE

can use a number. What makes values so powerful is that you can actually change them programmatically. For example, think of your score in a game like Angry Birds or the number of unread emails in your inbox. Both of these numbers are actually values. As you score more points in Angry Birds your score goes up; the value changes. As you read your email the number of unread emails you have goes down. Values are used to represent some kind of information.

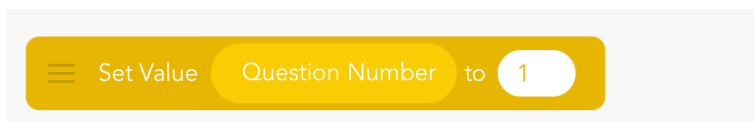
In Hopscotch, the values tab is to the right of the calculator tab that pops up when it's time to input a number. In addition to the built-in values like "iPad's width" or "Character's x position", you can add new values to your project to keep track of other things like a score or a state. It might be helpful to walk your students through the values menu and value blocks.

It is super important to name your values well! You'll need to be able to tell them apart later and remember what they're for!

After discussing the concept of values with your students, you might want to go through a list of objects that you will need and their respective initial rules (five text objects: a Title, a Question, and three Answers.) The title (or any object, really) should initialize a value that will keep track of what question the player is on. In the example, we call it "Question Number". We create separate values that will keep track of each answer category. We initialize "Question Number" to 1 and the three "Answer" values to 0.

## 4.1 Add a new rule to the Title object: Initialize Question Number to 1

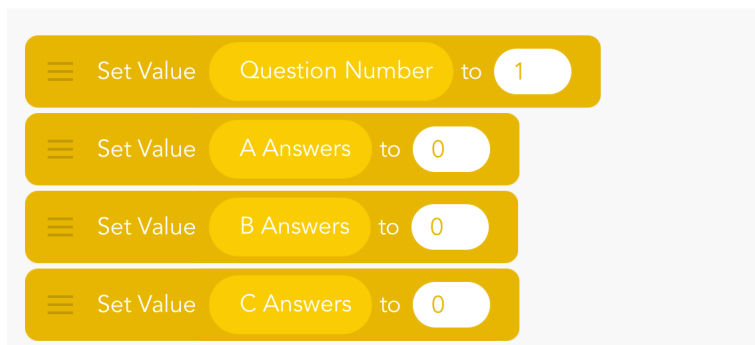
When the play button is tapped



When the values menu opens, tap the grey arrow on the top to go right to iPad's values. Create a new value with the button on the bottom of the menu. Call it "QuestionNumber".

## 4.2 Edit the Title object's rule: Initialize all three answer values to 0

When the play button is tapped



Create a new value for each Answer (e.g. A Answers, B Answers, C Answers)

# ACTIVITY GUIDE

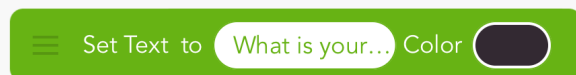
## 5. Value Events (10 minutes)

The four events at the very bottom of the Events menu deal with comparing values. They track whether a value is equal, not equal, bigger than, or smaller than another value or number. When you select the “\_ equals \_” event, the calculator menu opens. On the right of “Calculator” are values.

The quiz should display the first question and its answers when the “Question Number” value equals 1. The player chooses an answer and the game should keep track of how many of each kind of answer is chosen. We do that by increasing the appropriate answer value when each answer is tapped. For example, if an “A Answer” is chosen, the value “Answer A” should increase. Show the students these new events, and then discuss the rules that they will need to code before having them start programming. Make sure they choose the right object-value combination!

### 5.1 Add a new rule to the Question object

When 



*Type the first question in the “Set text” window.*

### 5.2 Add new rules to each of the Answer objects

When 



*Refer to your question matrix to enter each answer in its respective rule. We will do questions 2 and 3 later.*

### 5.3 Add new rule to each of the Answer objects: increment the right score when tapped

When 



When 





# ACTIVITY GUIDE

---

When

Answer C is Tapped



Increase Value

C Answers

by

1

## 6. Show Values (5 minutes)

It's hard to know whether our values are working properly, so programmers design tests to display them. In Hopscotch, you can do this by displaying the value on the screen. This is an advanced debugging strategy, and the values should be deleted from finished version of the project once you know everything works. Discuss this technique with students and see if they can figure out how to display a value when their project is played.

This is a good time to discuss **loops**.

Computers have a finite set of kinds of tasks they can accomplish. But when these tasks are combined properly, amazing things can be built. One thing they're particularly good at is repeating sets of instructions. In computer science we call this a "**loop**", or code that repeats.

Ask your students to imagine that they're baking cookies. They need to check on the cookies to make sure to take them out of the oven before they burn. You wouldn't just check once and then give up; rather, you would have to check every few minutes so that you catch them the moment that they're ready. The same is true of computers. We use loops to tell the computer to repeat actions for set periods of time (like checking cookie readiness).

As a class, discuss the behavior of the values in your quizzes and together make a list of the steps you want to happen. Ask students to consider the difference between using "Repeat 10 times" and "Repeat Forever". Which is appropriate for checking cookies? Which is appropriate for displaying the value?

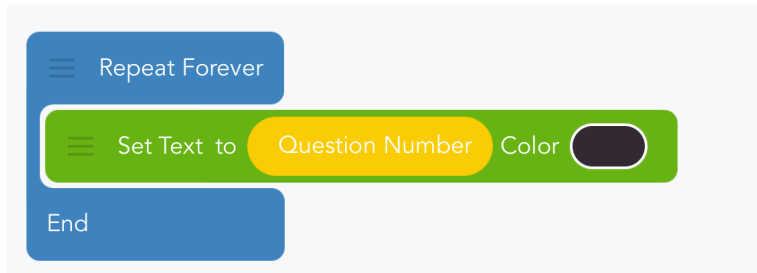
**6.1 Add four new blank text objects. Place them next to the Question and each of the Answers.**

# ACTIVITY GUIDE

---

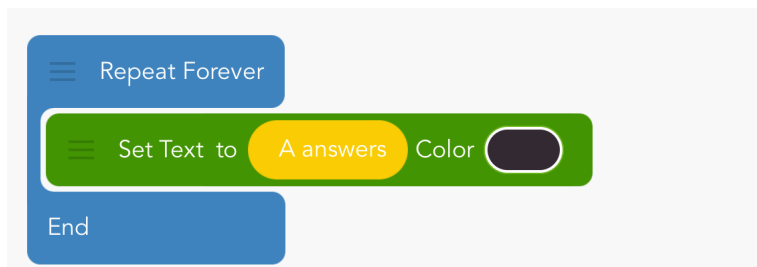
## 6.2 Add new rule to the Question object

When the play button is tapped



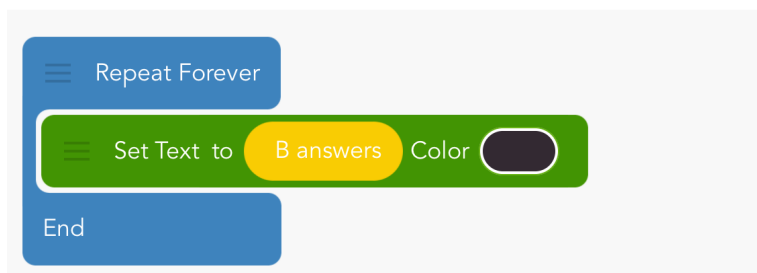
## 6.3 Add new rule to Answer A object

When the play button is tapped



## 6.4 Add new rule to Answer B object

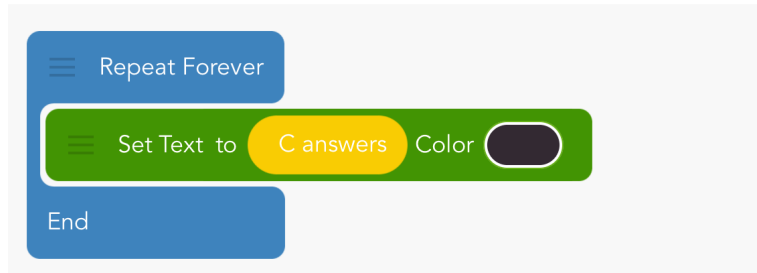
When the play button is tapped



# ACTIVITY GUIDE

## 6.5 Add new rule to Answer C object

When the play button is tapped



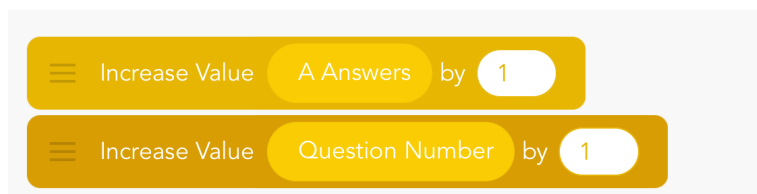
## 7. Next Question (5 minutes)

The score for each of the answers is advancing as you tap them, but the question does not change (neither does the "Question Number" value). Decide as a class, what event(s) should trigger the quiz move on to the next questions. If we make "Question Number" increase, we also need to make the question (and answers) display the correct text for when the "Question Number" value equals 2 or 3. How should this work?

Hint: You advance "Question Number" when any of the answers is tapped! We have already made rules for this event, so we just need to add a block to increase the "Question Number" value. You can do this step as a class and then together or in pairs test your programs to make sure the "Question Number" is advancing. Allow your students to implement the answer code in their own time and help their neighbors if they finish early.

## 7.1 Edit rules for all 3 answer objects to change "Question Number"

When Answer A is Tapped

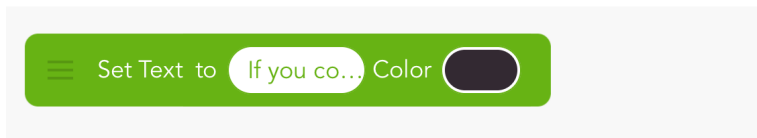


Add the Increase Value block to Answers B and C.

# ACTIVITY GUIDE

## 7.2 Add new rule for Question object to display second question

When 



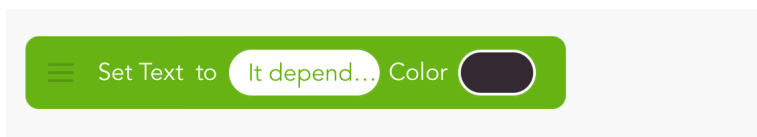
## 7.3 Add new rule for Question object to display third question

When 

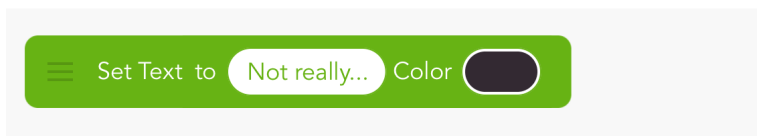


## 7.4 Add new rule for each Answer object to display second and third answers

When 



When 



*For Answer objects A, B, and C, add these same two rules with their respective answers. Refer to your matrix for the text of the answers. In this step, you're creating a total of 6 new rules!*

## 8. Results and introduction to Conditionals (5 minutes)

Now all the questions are answered, we have to check our results! The correct result to show is the one for which the "Answers value" is the greatest. We can use **conditionals** to compare values.

**Conditionals** are statements of the form "IF (something is true) THEN (do an action)". It executes code only under the *condition* that you specify, like IF the score is greater than 10 or IF a character is invisible. We use conditionals in our lives all the time. In the example of baking cookies, we used a conditional that said "If the cookies are done, **then** take them out of the oven." Conditionals are also extremely important in programming since computers need explicit directions.

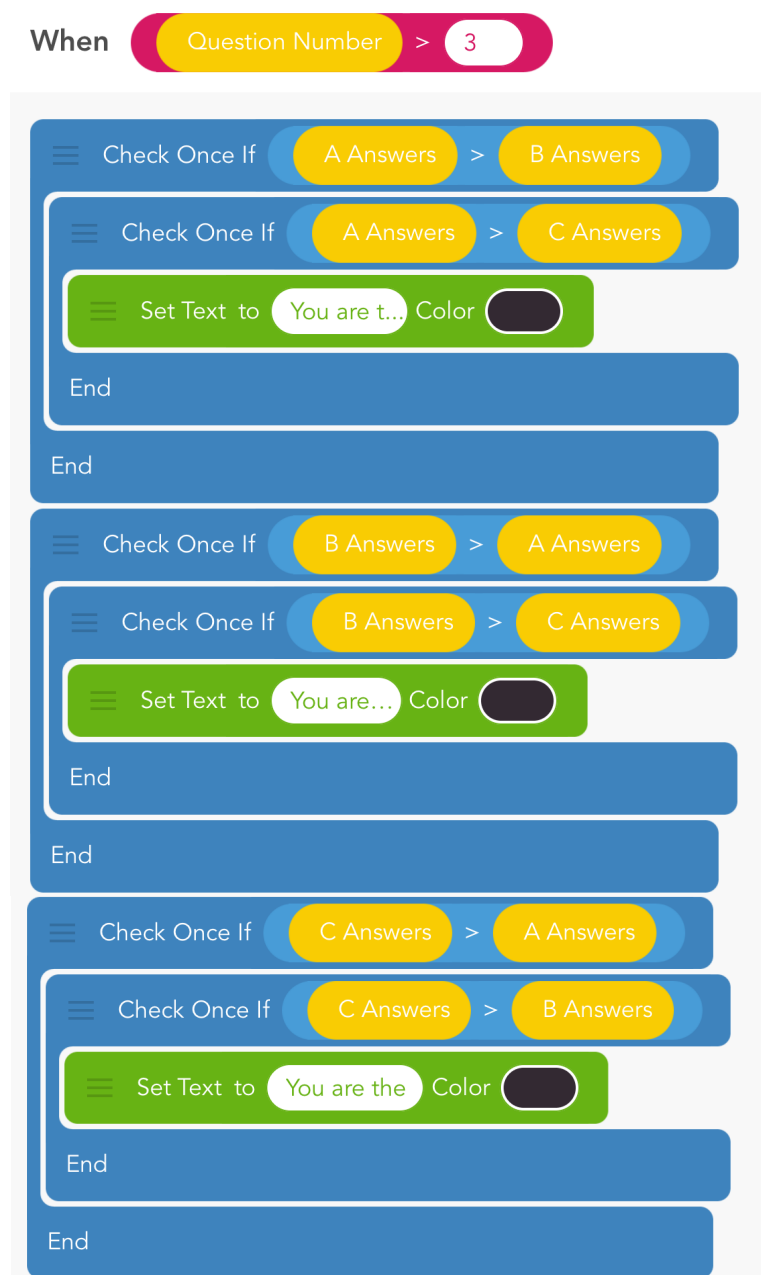
There are two types of conditionals in Hopscotch: "Check once if" and "Check if... else". "Check once if" is used when there are only two possibilities: 1. If the condition is true, do something, and 2. If the condition is not true, move on. "Check if else", by contrast, lets you check two things before moving on: 1. If something is true, do something, and 2. If this thing

# ACTIVITY GUIDE

is NOT true, do something else, then move on. For example, say you can go in the ocean only if you're wearing sunscreen or else you will have to put some on first. If you are wearing sunscreen (the condition is true), then you may go into the ocean. If you are not wearing sunscreen (the condition is false), then you must put some on first.

There are several ways to determine the answer to the quiz. The safest way is the one that won't have to be changed if more questions are added: A is the winner if the "A Answers" value is greater than *both* the "B Answers" and the "C Answers" values (and likewise for B and C.) In Hopscotch, we check if two things are true by nesting one inside the other.

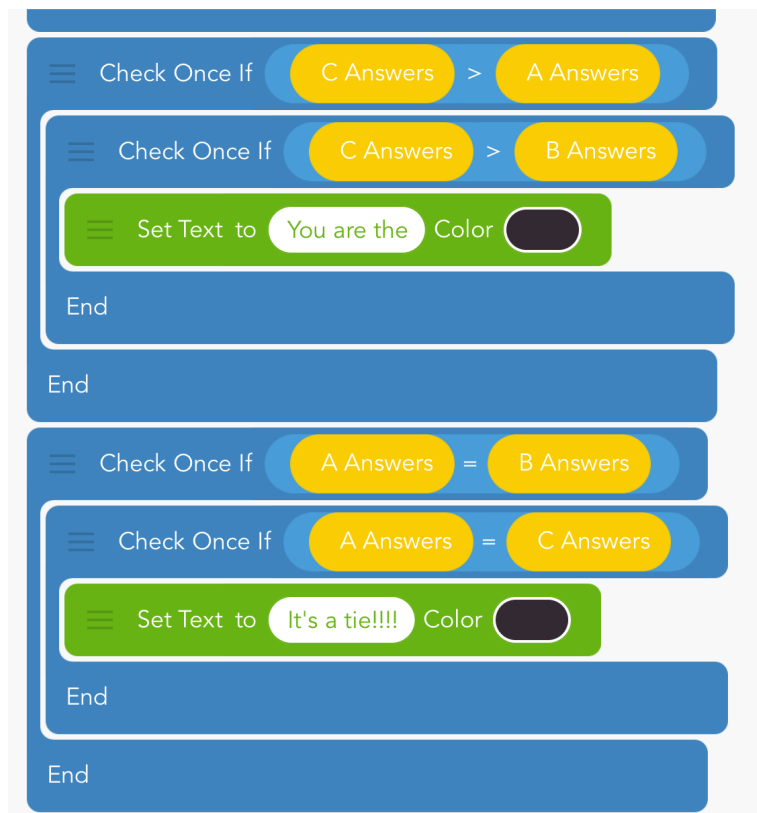
## 8.1 Add a new rule to the Title object



# ACTIVITY GUIDE

## 8.2 Edit the Title object's rule in case there's a tie (optional)

When Question Number > 3



*If you can't think of good result text for a tie, just set text to "It's a tie!"*

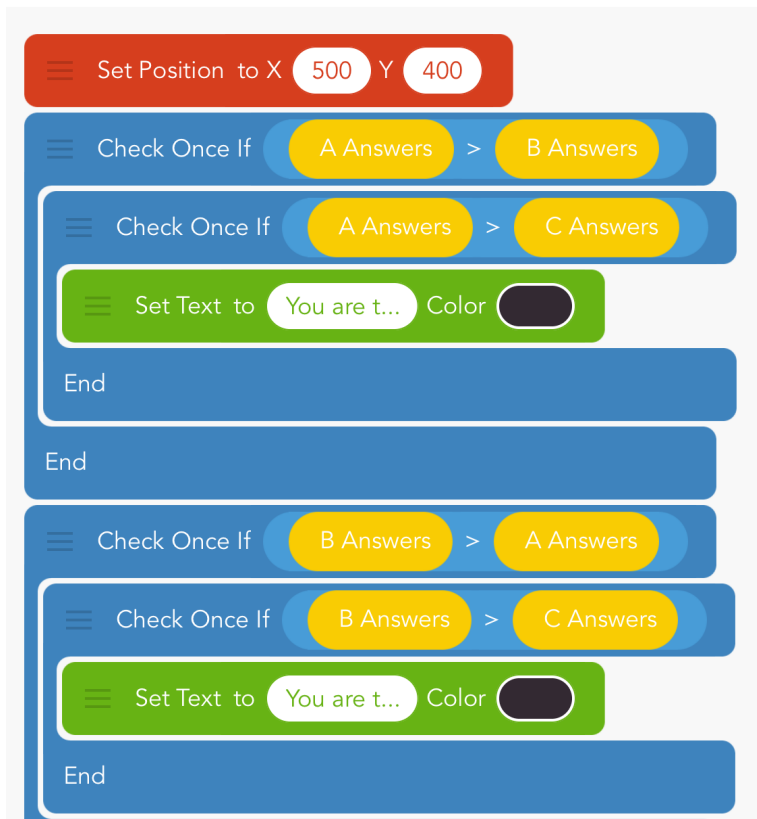
## 9. Tidying up (5 minutes)

The hard part is done, and there's just a bit more work to do to make the ending look snazzy. One option is to center the results and hide the question and answers at the end. Your students may come up with better ideas!

# ACTIVITY GUIDE

## 9.1 Edit Title's rule

When **Question Number** > **3**



A Scratch script for a 'When' trigger. The trigger is 'Question Number' > '3'. The script contains two main branches. The first branch starts with a 'Set Position to X 500 Y 400' block, followed by a 'Check Once If' block with 'A Answers' > 'B Answers'. If true, it runs another 'Check Once If' block with 'A Answers' > 'C Answers'. If true, it runs a 'Set Text to You are t... Color' block. The second branch starts with a 'Check Once If' block with 'B Answers' > 'A Answers'. If true, it runs another 'Check Once If' block with 'B Answers' > 'C Answers'. If true, it runs a 'Set Text to You are t... Color' block. Both branches end with 'End' blocks.

## 9.2 Add new rule to question and all three answers

*Don't forget to delete your test objects!*

When **Question Number** > **3**



A Scratch script for a 'When' trigger. The trigger is 'Question Number' > '3'. The script contains a single block: 'Set Invisibility Percent 100'.

## 9.3 Publish your project to the community for others to play and remix!

# DIFFERENTIATION

---

## (15 minutes, optional)

- How would you change the quiz to have more more answers or be longer?
- How could you change it to be a factual quiz—to change up whether the correct answer is A, B, or C?
- Turn the quiz into a survey that keeps track of how lots of people have answered—use this data to make a histogram.

# REFLECTION

---

## (5 minutes, optional)

- What is a value?
- What is a conditional?
- How are values and conditionals related or used together?
- How are events related to conditionals? How do you make a custom event? (When play button is tapped, repeat forever, check once if ...)
- What happens if there's a tie? How do we use conditionals to detect a tie? (if A=B=C)



# GLOSSARY FOR OLDER STUDENTS

---

**Ability/Function/Procedure/Subroutine:** A saved set of blocks. What we call abilities in Hopscotch are known as functions or subroutines in other programming languages. Easily replicable routines are a key concept in computer programming, and allow you to scale your code and create complex programs.

**Algorithm:** Algorithms are at the heart of computer science; they are the recipes that computers follow to solve problems.

**Bug:** An error that a programmer has made in their code

**Coding:** Writing the rules of behavior for a computer to follow automatically; programming

**Concurrency:** Two or more things happening at the same time, or triggered by the same event

**Conditional:** Statements of the form "IF (something is true) THEN (do an action)"

**Debugging:** Finding mistakes in your code (bugs) and fixing them

**Event:** A trigger that the computer recognizes and causes it to do some action. In Hopscotch, events include "When the iPad is tapped" or "When the play button is tapped"

**Iteration:** the repetition of a process

**Logic:** the science of the formal processes of thinking and reasoning

**Loop:** a repeating set of instructions

**Object:** A character or text with its own rules on screen

**Operator:** a mathematical symbol that produces a value

**Pair Programming:** a technique in which two people work together on one device. Hopscotch is developed using pair programming :)

**Program:** a set of instructions a computer can understand

**Programmer:** a person who writes programs

**Programming Language:** a set of words, rules, blocks or instructions that can be used to write a program.

**Random:** Any number or item among a set. The lack of a pattern among items in a set.

**Range:** The highest and lowest number random can choose between

**Rule:** Rules tell your object what to do and when to do it. When you make an ability and pair it with an event, you create a rule.

**Sequence:** An ordered list of things (instructions, blocks, numbers, etc) which can be triggered by an event or repeated

**Value:** A holder for a number. Also known as a variable

# GLOSSARY FOR YOUNGER STUDENTS

---

**Ability:** Code that can be reused

**Algorithm:** A recipe for a program

**Coding:** Telling computers what to do

**Concurrence:** Two things happening at the same time

**Conditional:** Statements of the form "IF (something is true) THEN (do an action)".

**Debugging:** Finding mistakes in your code and fixing them

**Event:** When something happens

**Iteration:** Having ideas and making mistakes, over and over

**Logic:** The process of making decisions

**Loop:** Code that repeats

**Operator:** A mathematical symbol that makes an equation

**Program:** A set of instructions a computer can understand

**Programmer:** A person who writes programs

**Programming Language:** A set of rules or blocks that can be used to write any program

**Random:** When there's no pattern

**Range:** The highest and lowest number random can choose between

**Rule:** Instructions that tell your computer what to do (the command) and when to do it (the event)

**Sequence:** The order in which instructions are given to the computer

**Object:** A character or text with its own rules

**Value/Variable:** A holder for a number