National Coding Symposium
Quorum Beginner Activity Lesson Plan

Compiled by Amanda Rodda

Audience:	This three lesson course is for anyone with basic computer skills: ● Can navigate to a website ● Can find links and buttons on websites ● Has basic typing skills You do not need to know anything about computer science or programming! If you know the basics of programming, you might want to try the intermediate course.

Expanded Core areas & components targeted in this lesson:

ECC Area	ECC Component

Learning targets (what do I want the students to learn?)

Objective #1	Learn about variables and user input while writing a Mad Lib about Louis Braille.
Objective #2	
Objective #3	

Vocabulary terms for this lesson:

- Algorithm- written instructions to complete a task
- Assignment Symbol- the symbol used to assign a value, in Quorum it is =
- Code- written instructions for a computer program
- Concatenation- linking two or more elements to achieve a result

- Input- information a computer gets from the user, devices, or other computers
- Output- information the computer gives a user, device or other computer
- Program- an algorithm that is run by a machine
- Variable- a label for information used in a program
    - Boolean- a variable that only holds true or false
    - Integer- a variable that holds integer values, no decimal points
    - Number- a variable that holds number values, can have decimals
    - Text- a variable that holds text values, words or symbols, always contained in quotation marks
- Adjective- describes a noun or pronoun
- Adverb- tells how something was done, usually ends in -ly
- Mad Lib- a word replacement game
- Noun- person, place, thing, or idea
- Verb- an action, something you do

Materials needed (What I need to teach the lesson?)

Teachers and students will need access to the doc files, and the Quorum website to actually code. No downloads are needed.

For this course, you will need any computer and a Quorum log in, so you can save your projects; access to the Quorum Language website, to use the online IDE, and the Google Doc lessons.

Quorum Log In

- Navigate to www.quorumlanguage.com.
- Once there, find the Login link.
    - For visual users, it is at the top of the page towards the right end of the list of links.
- That link will open a popup. Choose the Sign up for an account link.
    - Follow the directions to sign up for an account
- Once you have an account, if needed, sign in!
- You will learn how to save and load files once you make your first program.

Online IDE

An IDE is an integrated development environment. That is where you will write your code, find errors, and run your programs. There are several ways to access the Quorum online IDE. For this course, navigate to https://quorumlanguage.com/project.php. This page will allow you to use the IDE without other content getting in the way.

ECC High School Readiness Checklist sections referenced for this lesson:

Grade band:	Grades 5 - 8
ECC area: Assistive Technology	I can type at a minimum of 45 words per minute with 80% accuracy (five words per grade-- 45 by 9th, 50 by 10th, etc.). I can create, edit, save, and share a variety of formats such as Microsoft Word, Google Docs, Google Sheets, Excel, PDF, etc. I can use NVDA, JAWS, or Chromevox to complete word processing projects. I can use commonly used keyboard commands for reading, editing, and navigation.
Level: (Required, Novice, Accomplished or Proficient)	Required Skills

Lesson 1

Overview

In this lesson, you will learn about variables and user input while writing a Mad Lib about Louis Braille. Make sure you read the definitions before starting the lesson, so you will understand the vocabulary used.

Part 1: The basics

The first thing you need to know is variables. Just like math class, variables hold a value. Some change, and some stay the same throughout a program. Quorum has four type of variables:

- Boolean- a variable that only holds true or false
- Integer- a variable that holds integer values, no decimal points
- Number- a variable that holds number values, can have decimals
- Text- a variable that holds text values, words or symbols, always contained in quotation marks

For this program, you will use text variables. The code below shows how to write a text variable named word, and assign it the word thanks.

```
text word = "thanks"
```

What do you notice about the sample code?

You should always declare a variable by stating the type first. Then you give it a name. The name of the variable should make sense for what you are using it for. You don't want to name variables cat and dog, if they have nothing to do with a pet! Also notice that the variable name is lowercase. The = assigns the value. Since this is a text variable, the value thanks is in quotation marks. Now, copy the sample line, and paste it into the Quorum IDE, so retype it. Then use the Run button to run the program.

Note- Any time you copy and paste code from the lesson page to the IDE, you will have to delete and rewrite all quotation marks.

Did anything happen? Why?

Did you tell the computer to do anything with the variable?

No, you didn't.

New statements always go on a new line, so on a new line, add this:

```
output word
```

Before running the program, what do you think will happen?

If you look in the console, you will find two lines of text. (Visually, the gray box under the buttons. Screen reader, tab past Embed.) The first says "Build Successful." That means the computer was able to read all of the code. The second should be the word thanks. If you did not replace the quotation marks on thanks, you will have your first errors! Read them, and see if it makes sense that you should fix that word.

The last of the basics you need is user input. For our Mad Lib, we are going to use text input. The sample code below asks a question, saves it to a variable, and then outputs that variable. Enter this code and run it. Note- a popup will appear with the question. The output will always be in the console.

```
text answer = input("What is your name?")
output answer
```

What happened? Did the computer output a name, or the word answer? You use the variable's name to write the program, the computer uses the value when it runs.

Part 2: Let's DO something!

You know how to make a text variable, how to make the computer print to the screen, and how to get user input. Now let's write a Mad Lib. To start with, you will need a few sentences to work as the story part of the Mad Lib. Below is a short introduction to Louis Braille.

Louis Braille went blind at the age of three while playing with tools in his father's workshop. Louis first learned to read when he was ten years old. His first books were made of waxy paper with letters embossed into the pages. When Louis was 15, he took a new way to read that he learned from an Army chaplain, and improved it to become the dots we know as Braille.

Now you have a story. To make it a Mad Lib, you will take out 2 or 3 words per sentence, and replace them with a user input word. An example of words to remove, and how to write the variables and inputs is below. You are free to use these, or make your own!

Note- Using // makes the rest of the line a comment. That text will not be read by the computer when running the program. Commenting code is a great way to stay organized, and help others understand what you are doing.
Note 2- Clear your practice code, and start with a blank screen.

```
//Sentence 1
text number1 = input("Pick a number") //replacing three
text verb1 = input("Pick a verb ending in -ing") //replacing playing
text noun1 = input("Pick a place") //replacing workshop
//Sentence 2
text noun2 = input("Pick a name") //replacing Louis
text verb2 = input(Pick a verb") //replacing read
text number2 = input("Pick a number") //replacing ten
text time1 = input("Pick a timeframe, such as days, minutes, years")
//Sentence 3
text noun3 = input("Pick a plural noun") //replacing books
text adjective1 = input("Pick an adjective") //replacing waxy
text noun4 = input("Pick a plural noun") //replacing pages
//Sentence 4
text verb3 = input("Pick a past tense verb") //replacing learned
text verb4 = input("Pick a past tense verb") //replacing improved
text noun5 = input("Pick a noun") //replacing dots
text noun6 = input("Pick a language or way to write") //replacing Braille
```

That was a LOT to write! Let's take a moment to save your program. Use the Save button under the code editor. Name your project Louis Braille Mad Lib, then hit the Save Project button. DO NOT just hit enter after you name the project, it will eat your code!!

Now you will write a VERY long output statement, and use a concept called concatenation. Concatenation is when you add different parts of code together. In this case, you will add direct text to variables to produce the story. It will be much easier to copy the full story, the edit to add the concatenation. An example of the full output statement with concatenation is below.
Note- the text below has hard returns at the end of the line, as the IDE will not wrap text to a new line. This makes it difficult to read, and to find errors.

```
output "Louis Braille went blind at the age of " +  number1 + " while " +
verb1 + " with tools in his father's " + noun1 + ". " + noun2 +
" first learned to " + verb2 + " when he was " + number2 + " " +
time1 + " old. His first " + noun3 + " were made of " + adjective1 +
" paper with letters embossed into the " + noun4 +
". When Louis was 15, he took a new way to read that he " +
 verb3 + " from an Army chaplain, and " + verb4 + " it to become the " +
 noun5 + " we know as " + noun6 + "."
```

Run your code and see how it goes! Make sure to save your final changes, so you can share your code with others. In lesson 3, you will learn how to load a saved project.

In the completed example below, there is a second output that tells the correct story. You can add this to your code, or just leave it as the Mad Lib.

You can go on to Lesson 2.

Want to see the project all in one place, and run it?

Quorum Beginner Course
Lesson 2

Overview

In this lesson, you will learn about conditionals, and start writing an Escape Room program. Make sure you read the definitions before starting the lesson, so you will understand the vocabulary used.

Definitions

- Bug- an error in code
- Comparison Symbols- as in math, these are symbols that compare values.
    - Equal: =
    - Greater than: >
    - Less than: <
    - Greater than or equal to: >=
    - Less than or equal to: <=
    - Not equal to: not=
- Conditional Statement- a block of code that will only run when specific conditions are met

Lesson 2

Part 1: The Basics

In lesson 1, you learned how to make variables, concatenate elements, get user input, and use output statements. The first thing you will learn in this lesson is conditional statements. Conditionals are blocks of code that only run when specific conditions are met. In daily life, you might encounter a conditional as an if-then statement. Think about this comment from a teacher, “If you finish your work, then you may have free time.” The condition to be met is finishing your work. A program will skip over a conditional statement, if the condition is not true. Look at the sample code below, and think about what will happen, before you run it. Note- Remember to change the quotation marks if you are using copy and paste.

```
text name = “Angel”
        if name = “Omar”
                output “Hello”
        else
```

```
        output "I like fish."
    end
```

What did you notice about this code? A variable is made, and so is a conditional block. The conditional starts with if, and ends with end. All conditionals must have an end. You can put a comment after an end about what it is linked to. This can help you keep track of code blocks. Some programs have thousands of lines of code, and many nested blocks. A simple comment can save a lot of time when tracking down bugs. You should also notice that the "then" part of the conditional is just the statement on the next line. You do not label it with "then." This conditional also has an else in it. Else is used at the end of a conditional to give direction for any other input the computer may have.

The next sample conditional will use an integer variable, and will show that different kinds of comparison symbols can be used. Try and work through the logic before running the code. Also, please forgive the cheesy and awful output statements. Feel free to make your own!

```
integer age = 10
if age >=18
    output "Congratulations! You can vote!"
elseif age <= 6
    output "You're too young to go out alone."
else
    output "Behave in class!"
end //ends age conditional
```

What did you notice? Did you notice that the second conditional used elseif? That is the command used when checking multiple conditions. Did you notice the greater than or equal sign and less than or equal to notation?

The last concept you will learn in this lesson is nesting conditionals. You can have a conditional inside a conditional. You can also check more than one variable.

```
//This code is part of a card game simulator. Color refers to the color of the suit, and value would be the number on the card
text color = "red"
integer value = 10
if color = "black"
    if value < 10
        output "You get 1 point"
    else
        output "You lose 1 point"
    end //end nested value black
elseif color = "red"
    if value > 10
        output "You get 1 point"
    else
        output "You lose 1 point"
    end //end nested value red
end //end main color conditional
```

What do you think this will output? Did you notice the double ends at the end of the code? That is easy to forget!

Part 2: Let's DO something!

You are now going to start coding an Escape Room. Because you are using the online IDE, it will be easiest to write all statements as part of user input, so all text is contained in the popups, and the users won't have to bounce between the console and popup.

When using input statements with conditionals, every letter has to match. It is easiest to tell the user to write in either all uppercase or all lowercase. The example code will use lowercase, you are welcome to make your own choice.

The opening input should set the scene of the room, tell the user your choice of letter case, and ask a starting question.

Note- The sample code uses concatenation to break it onto two lines for easier reading, as discussed in lesson 1.

```
text name = input("You wake to find yourself in a dark room. " +
"While trying to find a way out, answer all prompts in all lowercase. " +
"First, what is your name?")
```

Saving this question as name allows you to use the user's name in the code, making it seem more personal, and more interactive. Next, you will add the call to action by simply asking the user if they want to look around. Again, this gives the game a more interactive feel. The sample code jumps to a new question if the player wants to continue, or out to the console, ending the game if they don't.

```
text start = (name + ", would you like to look around? yes or no?")
```

From here, you will add the shell of the main conditional statement. You will add to this shell in lesson 3, so don't worry if it doesn't run quite right. The example code chooses three objects to be in the room. As before, feel free to change the objects to make the game your own. The example is just there to help with formatting, and ideas.

```
if start = "yes"
        text startY = input("You see a chair, a table, and a bed. " +
        "Which would you like to look at? chair, table, or bed?")
        if startY = "chair"
                text chair = input("The chair has a rip in the cushion." +
                "Would you like to look inside? yes or no?")
        elseif startY = "table"
                text table = input("The top of the table is empty, "+
                "but you find a drawer underneath. Do you open the drawer?"+
                " yes or no?")
        else
                text bed = input("The bed looks unremarkable." +
                "Would you like to look underneath? yes or no?")
        end// end startY condition
else
        output "You are stuck in the room forever. Game Over!"
```

```
end //ends start condition
```

Make sure you save your program. Name it Escape Room, and remember to use the Save Program button, not just hitting enter after the name. You will load this program to finish it in the third lesson.

You can go on to Lesson 3.

Quorum Beginner Course
Lesson 3

Overview

In this lesson, you will learn how to make a counter, and finish the Escape Room program. Make sure you read the definitions before starting the lesson, so you will understand the vocabulary used.

Definitions

- Iteration- repeating code
- Loop- a block of code that is run multiple times, in Quorum, loops end with an end
    - repeat x times- this is the format to repeat a block of code a specific number of times
    - repeat while- this block will repeat as long as the condition is true
    - repeat until- this block will end when a condition is met

Lesson 3

Part 1: The Basics

For this lesson, you will need to load your Escape Room project. Navigate to the My Projects link. Visually, this is at the top of the screen near the Log Out link. On this screen you will find a list of saved projects. Checking the public box allows you to share a link to your code. The share button will give you the link. You don't need that right now. Navigate to the Load button. This will take you to an IDE with your code. You will also notice an Embed button. If you have your own website, you can embed a Quorum IDE into it, with the code to your project, allowing others to run and play your game.

You will add a loop to your program to track the player's progress, and know when to end the game. Loops are blocks of code, like conditionals. The sample code below uses a loop to count by 2s. See if you can figure out the output before running the code.

```
integer countBy2 = 0
repeat 5 times
        countBy2 = countBy2 + 2
        output countBy2
end //end repeat
```

The next sample code will use a counter to track iterations, and an until loop using that counter.

```
integer counter = 0
repeat until counter = 3
        output "The counter = " + counter
        counter = counter + 1
end //end repeat
```

Part 2: Finish the Escape Room

The next step will be to make a counter to track when the player checks a location. The example code will have the win condition be when the player checks all three locations. You are encouraged to write your own exploration and way to escape. The counter will be made at the top of the code. It will then increment the counter when the player chooses to search an item. A loop will be added after the counter, and ended at the end. The sample code will also finish the nested conditionals needed for searching. This is a lot of frontloaded explanations, but there will also be comments in the code to help explain.
Note- return now is used to exit conditionals or loops and end the program
Note 2- some else statements were left blank to exit the conditional
Note 3- Another conditional was added at the end, when the counter is 3, to end the game.

```
integer searches = 0 //this is the counter
text name = input("You wake to find yourself in a dark room. " +
  "While trying to find a way out, " +
  "make sure you answer prompts in all lowercase. " +
  "First, what is your name?")
text start = input(name + ", would you like to look around? yes or no?")
repeat until searches = 3
   if start = "yes"
      text look1 = input("You see a chair, a table, and a bed. " +
        "Which would you like to look at?" +
         " chair, table, or bed?")
         if look1 = "chair"
            text chair = input("The chair has a rip in the cushion. " +
              "Would you like to look inside? yes or no?")
               if chair = "yes"
                  text chairY = input("In the chair cushion you find a magnet." +
                     " Press enter to continue your search.")
                  searches = searches + 1 //this increments the counter
               else
                 //left blank to exit the conditional
               end //ends chair conditional
        elseif look1 = "table"
            text table = input("The top of the table is empty, " +
              "but you find a drawer underneath. Do you open the drawer?" +
              " yes or no?")
              if table = "yes"
```

```
                text tableY = input("You find the bow, or head, of a key." +
                    " Press enter to continue your search.")
                searches = searches + 1 //this increments the counter
            else
                //left blank to exit the conditional
            end //ends table conditional
        else
            text bed = input("The bed looks unremarkable." +
                " Would you like to look underneath? yes or no?")
                if bed = "yes"
                    text bedY = input("You lift the mattress and find the blade," +
                        " or teeth, of a key. Press enter to continue your search.")
                    searches = searches + 1 //this increments the counter
                else
                    //left blank to exit the conditional
                end // ends bed conditional
        end //ends look1 conditional
    else //this is part of the start conditional from the beginning
        text done = input("You are stuck in the room forever. Game Over!")
        return now //this exits the conditional and ends the program
    end //end start
end //end loop

//This loop runs after the search, and wraps up the game
if searches = 3
    text ending = input("You have found three items. Do you want to try and magnetize " +
        "the key back together? yes or no?")
    if ending = "yes"
        text done = input("It worked! You are able to escape the room! ‟ +
            ‟Congratulations!")
    else
        text done = input("You were so close, but are stuck in the room forever! " +
        "Game Over!")
        return now //this exits the conditional and ends the program
    end // ends ending conditional
end //ends searches conditional
```

Congratulations! You have completed the APH Coding Symposium Quorum Language Beginner's course!

Want to see the final project and run it?