LESSON PLAN: Demonstrating Change Over Time Using Moon Phases

Unit Plan Overview

Description
Students create/modify a computer program in Scratch* where they demonstrate their knowledge of the different moon phases while learning five computer programming and computational thinking concepts. Students apply their knowledge of the science of the different phases of the moon.

*As the educator, having some familiarity with Scratch is important. If you are unfamiliar with the Scratch environment, you can watch the following video.

Grade Levels	**Scratch Level**	**Duration**
3-5	Novice to proficient	Novice coders: 240 minutes Proficient coders: 180 minutes

Educational Outcomes
Students will:
- Create or modify a computer program to represent the phases of the moon for a month.
- Explain lunar patterns using a model they created in Scratch.

Standards Connections
Bolded parts of the standards are fully met by this lesson.

California Computer Science State Standards

***3-5.AP.12* Create programs that include events, loops,** and conditionals.

***3-5.AP.13* Decompose problems into smaller, manageable tasks which may themselves be decomposed.**

3-5.AP.14 Create programs by incorporating smaller portions of existing programs, to develop something new or add more advanced features.

3-5.AP.17 Test and debug a program or algorithm to ensure it accomplishes the intended task.

NGSS DCI

ESS1.B Earth and the Solar System.

Standards connections in part C:

***NGSS 5-ESS1-2** ***Represent data in graphical displays to reveal patterns of daily changes** in length and direction of shadows, day and night, and the seasonal appearance of some stars in **the night sky.**

***NGSS Performance Expectation notes specify the inclusion of moon patterns.**

21st Century Skills

***Collaboration*: Exercise flexibility** and willingness to be helpful in **making necessary compromises to accomplish a common goal.** (http://www.p21.org Collaboration 2).

Programming and Computational Thinking Concepts

Programming:

- Sequencing - An order of events that happens logically.
- Conditionals - If the condition is true, then the follow-up code will proceed, but only if the condition is true.
- Loops - The code within the "loop" will repeat for a set number of times - the code for this lesson uses the "forever" loop of Scratch.
  - Note: In programming there is no "forever" loop, but usually a set number of repetitions. In Scratch there could be 1000 repetitions which may make it seem like "forever."
- Operators - Perform logical and mathematical operations.
- Variables - A value that is stored and can change; in this lesson, the dates for a calendar.

Computational Thinking:

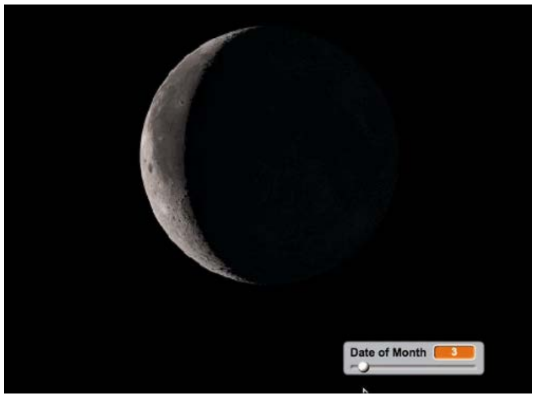
- Decomposition - Breaking down problems into manageable tasks.
- Pattern recognition - Recognizing if there is a pattern and what the sequence is.

Set-Up and Prep

Observe and record moon phases for a month by studying the moon in the night sky every evening.
If this is not feasible, or if student handouts are incomplete, use a moon phase generator to make sure that all students have a completed and correct moon phase calendar for the same month.

Materials (per class of 32 students)

Devices with Wi-Fi	Handouts
• 12-16 devices to share (2-3 to a device). • Laptops, Chromebooks and tablets will work.	• Flowchart • Moon calendar • Example challenge

Scratch Programs	GIF of Program
• Change Over Time - Full program • Skeleton Change Over Time - Partial program	

Lesson

A. Program Introduction (60-120 minutes)

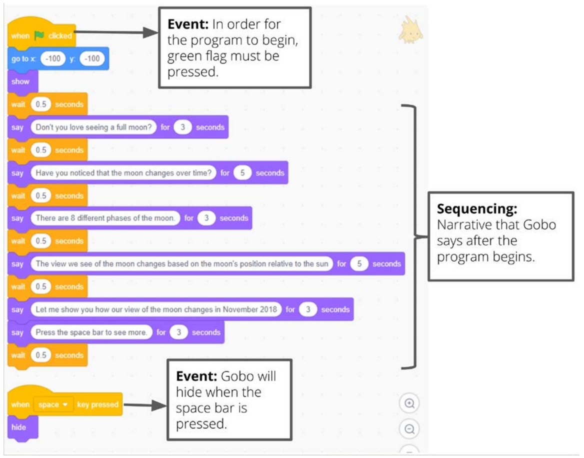
1. **Questioning.*** Students run Change Over Time full program. Focus on the output of the program.
   - Make observations on the output and use the following questions to guide students observations:
     - Describe parts of the program. What are the different pieces that make up this program?
     - Identify where you see patterns throughout the program.
     - How are the different parts of the code initiated? What do you need to do to initiate them? How are certain parts of the program triggered?
     - Where do you notice repetition in the program?

2. **Concept Connections.*** Introduce the program code to students.
   - Observe the following computational thinking and programming concepts. (See the screenshots below for the locations of computational thinking and programming concepts.)
     - Decomposition — Describe parts of the program. What are the different pieces that make up this program?
     - Pattern recognition — Identify where you see patterns throughout the program.
     - Events — How are the different parts of the code initiated? What do you need to do to initiate them? How are certain parts of the program triggered?
     - Loops — Where do you notice repetition in the program?

   *Note: Questioning and Concept Connections can be combined to ensure students see the connection between questioning and computing concepts.

3. Below are additional ways to engage students in having a deeper understanding of the program code.
   - Provide screenshots of the code and have students decipher and predict what the output will be (see below).
   - Introduce programming vocabulary and have students identify where each concept is being demonstrated and how that's an example of that concept.
   - Briefly review a Sprite's code and the programming concepts present. Next, have students review the code for the remaining Sprites individually or in small groups. Once completed, discuss the class findings for how code is functioning for the remaining Sprites.

Phases of the moon program — screenshots:

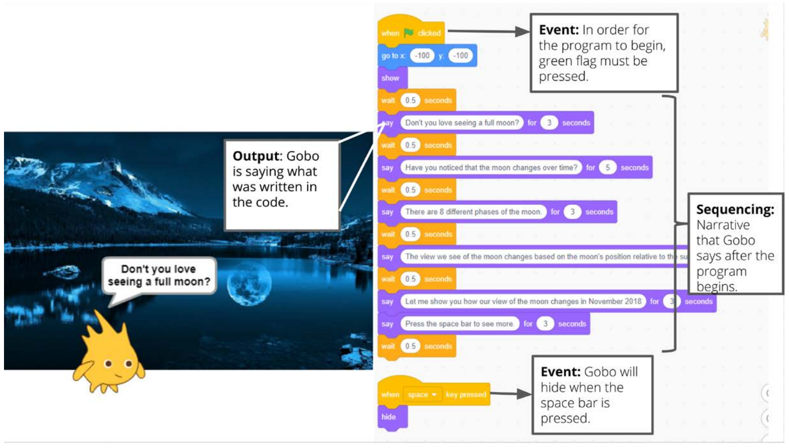


```
when [flag] clicked
go to x: -100 y: -100
show
wait 0.5 seconds
say Don't you love seeing a full moon? for 3 seconds
wait 0.5 seconds
say Have you noticed that the moon changes over time? for 5 seconds
wait 0.5 seconds
say There are 8 different phases of the moon. for 3 seconds
wait 0.5 seconds
say The view we see of the moon changes based on the moon's position relative to the sun for 5 seconds
wait 0.5 seconds
say Let me show you how our view of the moon changes in November 2018 for 3 seconds
say Press the space bar to see more. for 3 seconds
wait 0.5 seconds

when space key pressed
hide
```

Event: In order for the program to begin, green flag must be pressed.

Sequencing: Narrative that Gobo says after the program begins.

Event: Gobo will hide when the space bar is pressed.



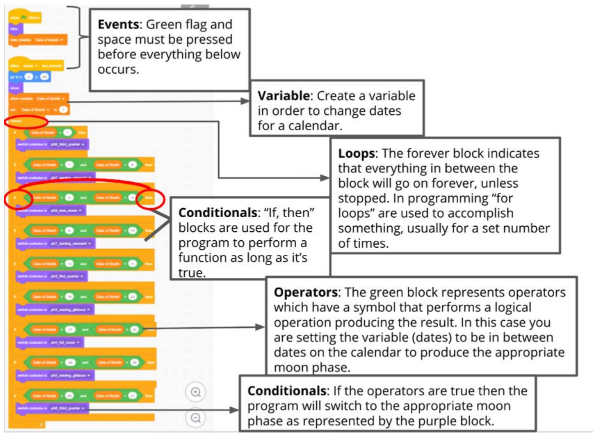
```
when [flag] clicked
go to x: -100 y: -100
show
wait 0.5 seconds
say Don't you love seeing a full moon? for 3 seconds
wait 0.5 seconds
say Have you noticed that the moon changes over time? for 5 seconds
wait 0.5 seconds
say There are 8 different phases of the moon. for 3 seconds
wait 0.5 seconds
say The view we see of the moon changes based on the moon's position relative to th
wait 0.5 seconds
say Let me show you how our view of the moon changes in November 2018 for 3 seconds
say Press the space bar to see more. for 3 seconds
wait 0.5 seconds

when space key pressed
hide
```

Event: In order for the program to begin, green flag must be pressed.

Sequencing: Narrative that Gobo says after the program begins.

Event: Gobo will hide when the space bar is pressed.

Output: Gobo is saying what was written in the code.



Events: Green flag and space must be pressed before everything below occurs.

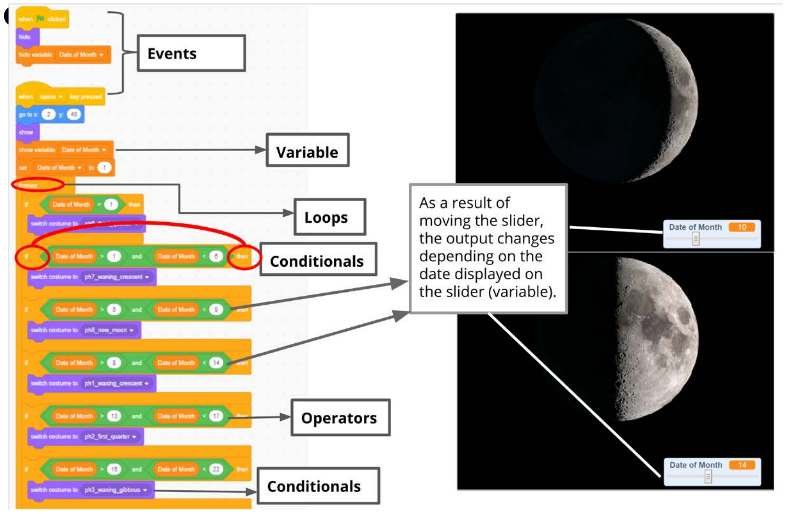
Variable: Create a variable in order to change dates for a calendar.

Loops: The forever block indicates that everything in between the block will go on forever, unless stopped. In programming "for loops" are used to accomplish something, usually for a set number of times.

Conditionals: "If, then" blocks are used for the program to perform a function as long as it's true.

Operators: The green block represents operators which have a symbol that performs a logical operation producing the result. In this case you are setting the variable (dates) to be in between dates on the calendar to produce the appropriate moon phase.

Conditionals: If the operators are true then the program will switch to the appropriate moon phase as represented by the purple block.



As a result of moving the slider, the output changes depending on the date displayed on the slider (variable).

B. Design Challenge (60-120 minutes)
This challenge allows students to develop a deeper understanding of how the program works and what effect each block has on the output.

1. Have students familiarize themselves with the Change Over Time full program by having them explore the code and practice interacting with the environment.

2. Ask students: *What are some ways we can modify the Change Over Time program? Ideas for determining criteria?* (Middle school only):
   - Change how the program is initiated.
   - Add additional blocks to enhance the program (e.g., students can add sound after a specific action).
   - Change some or all of the Sprites to represent the different moon phases.
   - Change the variable maximum to include multiple months.

3. Sample design challenge:

Frame the Challenge Modify the Change Over Time moon program as outlined in the criteria below.
Criteria (Design Requirements/Desired Features) • Change at least two costumes to different Sprites — e.g., students can use a banana Sprite for a crescent moon and a baseball Sprite for a full moon. • Change values of the operators. • Add or remove a conditional block. • Each team member must contribute and make at least **one** change to the program.
Constraints (Design Limitations) • 30 minutes to make modifications. • Work with your assigned group.
Testing • Test your program after every modification and observe what happens.

4. Questions to facilitate sharing solutions:
   - How does your program differ from the original moon phases program?
   - How is your program the same as the original moon phases program?
   - Where in the program did you switch costumes?
   - What does your program do? How does it do that?

C. Content Connections

1. Ensure that all students have a complete and correct moon phase calendar for a specific month. See set-up instructions for more details.

2. Introduction to identifying patterns:
   - Tell students: *The program we have been exploring models how the moon does not always look the same when we observe it at night; however, there are some patterns in how it will appear from one month to the next. Using your observations from your complete Moon Calendar handout*, compare your data to those from the month of November 2018 and see if there are any similarities or differences.*
     - *Describe what you see in your Moon Calendar handout.*
     - *Describe what you see in the November 2018 moon calendar.*
     - *Describe any patterns that you observe in your Moon Calendar handout or in the month of November 2018.*
     - Students should realize that every month has a full moon and a new moon, but each may occur at different times of the month.
   - Based off data in student Moon Calendar handout* and the month of November, students can identify patterns to the phases of the moon. These patterns include:
     - Eight phases within a 28-day cycle (new moon, waxing crescent, first quarter, waxing gibbous, full moon, waning gibbous, third quarter, waning crescent).
     - Each phase lasts about 3-4 days.
     - The phases always cycle through the same pattern, though the start point at the beginning of any given month will change.

3. Connecting the design challenge with content:
   - *How are the patterns of the moon phases represented by the Change Over Time program?*
   - *How are the patterns in the code for the program similar to or different from the moon phase patterns?*

*If you are unable to complete the Moon Calendar handout, the following resource can be used in its place.

D. Iterate Design Solution

In this section, students create a moon phases program (by either modifying existing code or creating a new program).

Brainstorming: As with any program, have students create a pseudocode or flowchart to organize their thinking before they begin programming. This is especially useful for students who are new to programming and may need to write out their steps before coding them.

Real-world connection: Software engineers usually don't start from ground zero; they normally take existing code (written by them or by other programmers) and modify it to solve their problem. When this is done, it's important that credit is given to the developer (see "Notes and Credits" section in Scratch).

1. Depending on your students' exposure and knowledge of Scratch, there are multiple ways to complete the design challenge.

Novice coders	Proficient coders
• Provide students with the full program and have them modify it to fulfill the design challenge. • Provide a guided handout and a program where there is some basic code in certain or all of the Sprites.	• Have students create their own moon phase program (without starting from a sample) — Sprites and code are not provided. • Provide all of the Sprites students can use, but don't have any code attached to them — students decide the code and how to use each Sprite. Challenge criteria could be that students have to use all of the Sprites provided.

2. Questions for determining criteria and constraints (Middle school only):
   - What do you want your program to do?
   - How is your program going to accomplish that?
   - What type of blocks do you think you will need to use?

3. Design challenge, including iteration details:

Frame the Challenge *You are applying for a programming position at NASA Ames. Before they can hire you they want you to complete a performance task: Create a simple moon phase program showing all phases of the moon and explain your thought process.*
Criteria (Design Requirements/Desired Features) • Use at least five blocks. • Use at least three different types of blocks (motion, looks, etc.). • Show all phases of the moon represented by your data.
Constraints (Design Limitations) • 30 minutes to complete the challenge. • Use Scratch as the platform for your moon phase program. • Use the materials provided (computer, handout(s), etc.).
Testing • Test your program often and test each section to ensure that it functions as you intended.

4. Once students have completed the challenge, it is important they share their program with others. Have students run the program and talk through what is happening using the programming concepts. Questions to facilitate sharing solutions:
   - What does your program do? How does it do that?
   - Did your program work the first time you coded/wrote it? Why or why not? (Here students would hopefully talk about how they had to debug their program in order for it to run).
   - Are there other ways to code your program? (Students eventually realize that there are many ways to code a program that does the same thing.)
   - Is your program efficient? (This is a higher level question where students realize that there are more efficient ways to write code - nesting is not always the answer.)
   - How does this program help you explain changes over time?

E. Evaluation

This section summarizes suggestions for implementing summative evaluations, as well as creating authentic experiences for the students around the design challenges and learning by making work public.

1. Introduction
   - *Now that you have shared your program with others, let's explain your thought process in creating the program.*
   - *Remember that you want the position as a computer programmer at NASA!*

2. Introduce and explain the final assessment project with students:
   - *Using evidence, explain how you created your moon phase program.*
   - *You will run your computer program and present your performance task to the NASA hiring manager and explain how you will be an asset if hired.*

3. Questions to connect the design challenge to the assessment project and/or project goals:
   - *Looking through your moon phases program and that of others, what did you notice?*
   - *Describe the differences observed.*
   - *Describe the similarities observed.*

Appendix A – Grade Level Modifications

Modifications for all grade levels
The moon phases program models change over time. After completing this lesson, students can use their programming skills to create a model for other content concepts related to change over time. It is important to provide time to research content, collect their evidence, create a program and share out. Below are standard alignments for various grade levels.

Grade	Topic	Standard
3	Organism life cycles (animal or plant)	**3-LS1-1** Develop models to describe that organisms have unique and diverse life cycles but all have in common birth, growth, reproduction and death.
4	Erosion	**4-ESS2-1** Make observations and/or measurements to provide evidence of the effects of weathering or the rate of erosion by water, ice, wind or vegetation.
4	Mission expansion	**4.2.4** Describe the mapping of, geographic basis of, and economic factors in the placement and function of the Spanish missions, and understand how the mission system expanded the influence of Spain and Catholicism throughout New Spain and Latin America.
5	Trade routes	**5.2.3** Trace the routes of the major land explorers of the United States, the distances traveled by explorers, and the Atlantic trade routes that linked Africa, the West Indies, the British colonies and Europe.
6	Trade routes	**6.7.3** Identify the location of and the political and geographic reasons for the growth of Roman territories and expansion of the empire, including how the empire fostered economic growth through the use of currency and trade routes.
7	Trade routes	**7.2.5** Describe the growth of cities and the establishment of trade routes among Asia, Africa, and Europe, the products and inventions that traveled along these routes (e.g., spices, textiles, paper, steel, new crops), and the role of merchants in Arab society.
7	Population changes	**7.6.7** Map the spread of the bubonic plague from Central Asia to China, the Middle East, and Europe and describe its impact on global population.
8	Westward expansion	**8.5.2** Know the changing boundaries of the United States and describe the relationships the country had with its neighbors (current Mexico and Canada) and Europe, including the influence of the Monroe Doctrine, and how those relationships influenced westward expansion and the Mexican-American War.
6-8	Describing cyclic patterns - eclipses, seasons, moon phases	**MS-ESS1-1** Develop and use a model of the Earth-sun-moon system to describe the cyclic patterns of lunar phases, eclipses of the sun and moon, and seasons.
6-8	Population increase	**MS-ESS3-4** Construct an argument supported by evidence for how increases in human population and per-capita consumption of natural resources impact Earth's systems.

6-8	Transfer of energy - photosynthesis, cycling of matter in an ecosystem	**MS-LS1-6** Construct a scientific explanation based on evidence for the role of photosynthesis in the cycling of matter and flow of energy into and out of organisms.
6-8	Erosion	**MS-ESS2-2** Construct an explanation based on evidence for how geoscience processes have changed Earth's surface at varying time and spatial scales.
6-8	Ecosystems	**5-LS2-1** Develop a model to describe the movement of matter among plants, animals, decomposers and the environment.
6-8	Character development	**CCSS.ELA-Literacy.W.6.3a/b** Use narrative techniques, such as dialogue, pacing, and description, to develop experiences, events and/or characters.
9-12	Transfer of energy - cellular respiration	**HS-LS1-7** Use a model to illustrate that cellular respiration is a chemical process whereby the bonds of food molecules and oxygen molecules are broken and the bonds in new compounds are formed, resulting in a net transfer of energy.
9-12	Life cycle of stars	**HS-ESS1-3** Communicate scientific ideas about the way stars, over their life cycle, produce elements.
9-12	Character development	**CCSS.ELA-Literacy.RL.9-10.3** Analyze how complex characters (e.g., those with multiple or conflicting motivations) develop over the course of a text, interact with other characters, and advance the plot or develop the theme.

Appendix B – Vocabulary

Below are a suggested list of words to discuss as you progress through this unit with students. For more in depth information about vocabulary and teaching information, see Tech Tip: The Language of Engineering.

Term	Student-friendly definition
Computational Thinking	A problem-solving process that uses logic and computers or other tools to solve a problem.
Condition	Only happens if the statement is true.
Event	An action that needs to happen for the code to run.
Input	Data a computer receives.
Loop	A set of instructions that will happen over and over for a set amount of times.
Operator	A math symbol that performs a math action, e.g. 3*3, the * would multiply both numbers.
Output	What is displayed on a screen of a computer based on the information given to the computer.
Sequence	The order in which the code will happen.
Variable	A value that can change depending on what is happening.

Appendix C - Resources and References

Argument Learning Materials

- "Moon Calendar December 2018." *Calendar-365*. www.calendar-365.com/moon-calendar/2018/December.html.

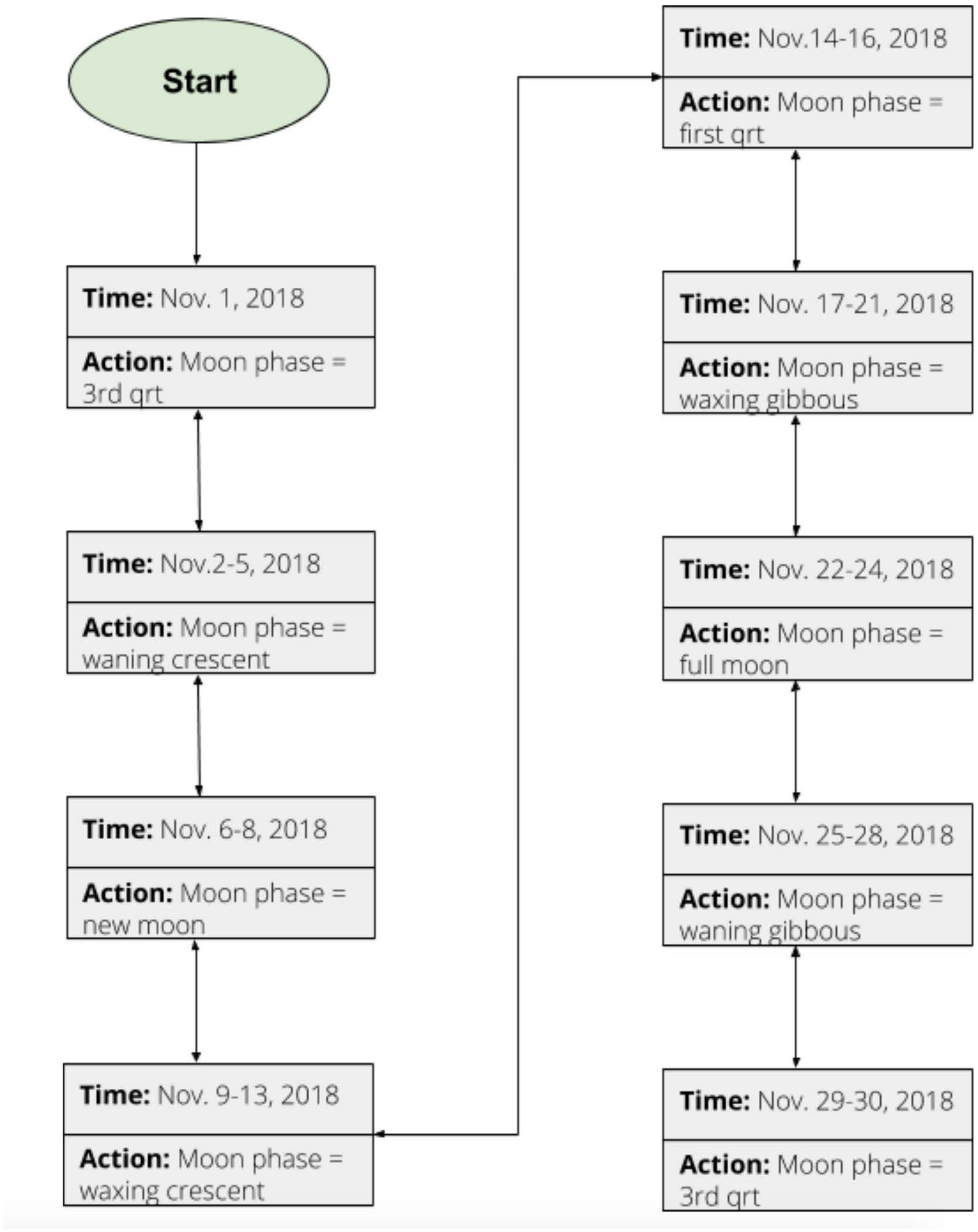
- "Moon Calendar November 2018." *Calendar-365*. www.calendar-365.com/moon-calendar/2018/November.html.

- West, Richard. "Scratch 3 Beginner's Tutorial - Beetle Game (Scratch 2019)." *YouTube*, uploaded by Learn Learn Scratch Tutorials, 3 Jan. 2019, www.youtube.com/watch?v=CAccFtzGF_g.

- Wright, Ernie. "Moon Phases Loop." *NASA*, 2015, svs.gsfc.nasa.gov/4310.

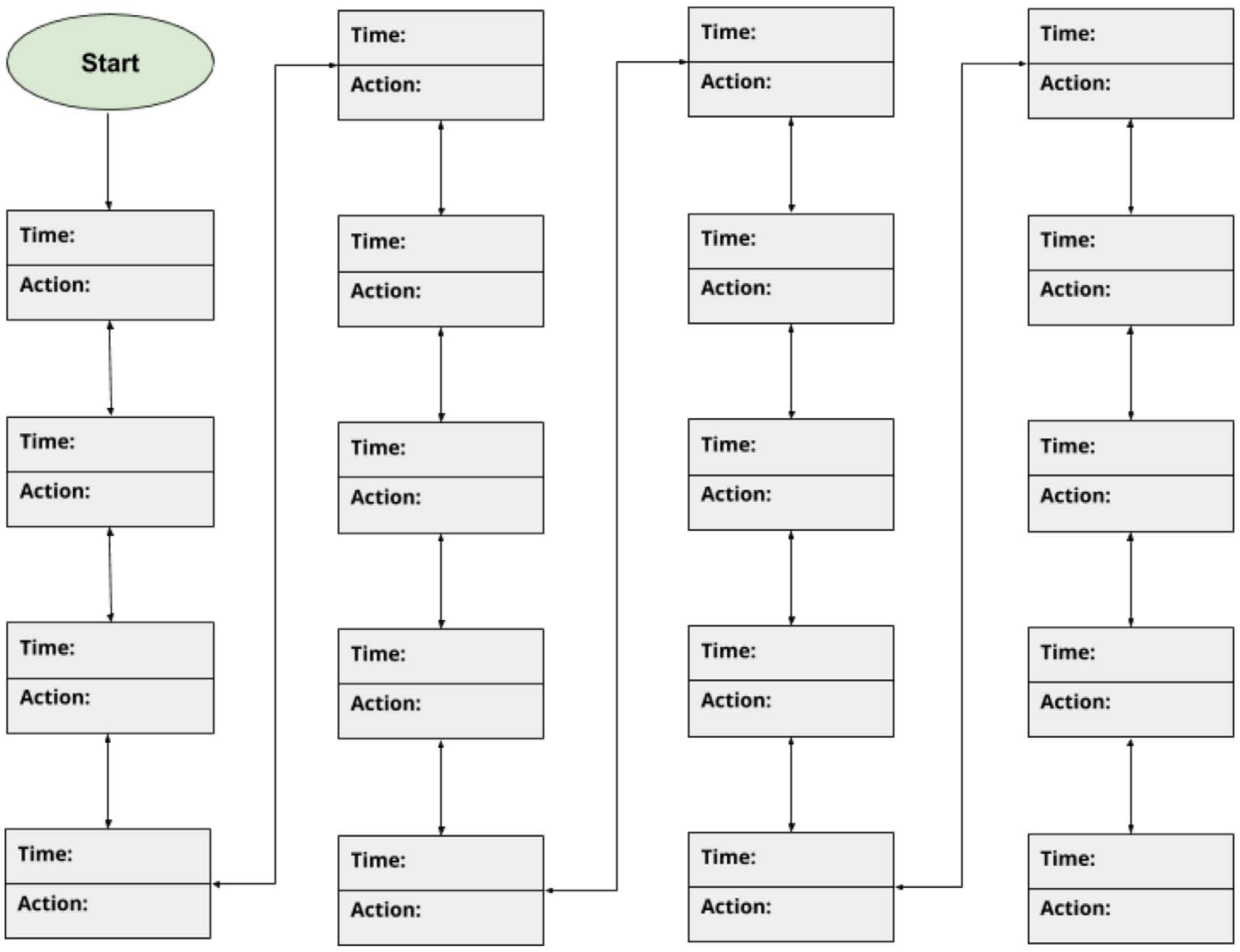
Appendix D - Lesson Handouts

Change Over Time Flowchart Example*


Start
Time: Nov. 1, 2018
Action: Moon phase = 3rd qrt
Time: Nov.2-5, 2018
Action: Moon phase = waning crescent
Time: Nov. 6-8, 2018
Action: Moon phase = new moon
Time: Nov. 9-13, 2018
Action: Moon phase = waxing crescent
Time: Nov.14-16, 2018
Action: Moon phase = first qrt
Time: Nov. 17-21, 2018
Action: Moon phase = waxing gibbous
Time: Nov. 22-24, 2018
Action: Moon phase = full moon
Time: Nov. 25-28, 2018
Action: Moon phase = waning gibbous
Time: Nov. 29-30, 2018
Action: Moon phase = 3rd qrt

*Notice that there are **nine** Time/Action blocks and there are also **nine** conditional blocks ("if, then" blocks) in the Scratch file.

Change Over Time Flowchart



Moon Calendar

[Month, Year]						
Sunday	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**
[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]
[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]
[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]
[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]
[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]
[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]	[Day] [Phase]

Example Challenge Worksheet: Moon Phases

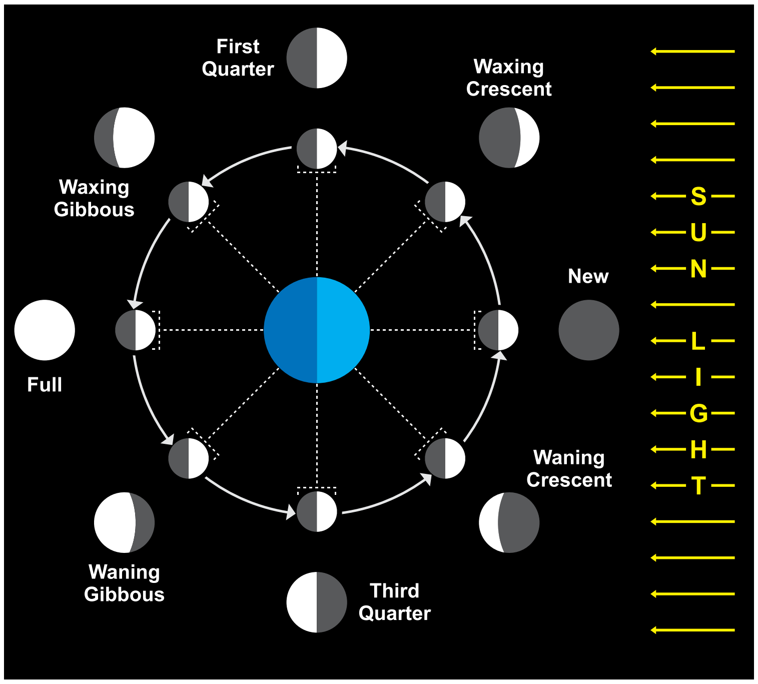
Challenge: You are applying for a programming position at NASA Ames. Before they can hire you they want you to complete a performance task: Create a simple moon phase program showing all phases of the moon and explain your thought process.

Criteria	Constraints
• Use at least five blocks. • Use at least three different types of blocks (motion, looks, etc.). • Must show all phases of the moon represented by your data.	• 30 minutes to complete the challenge. • Use Scratch as the platform for your moon phase program. • Use the materials provided (computer, worksheet, etc.).

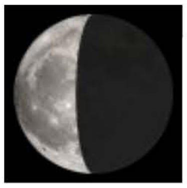
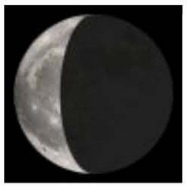
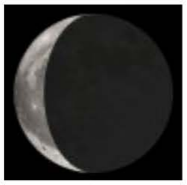
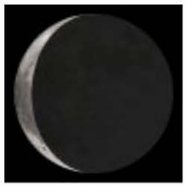
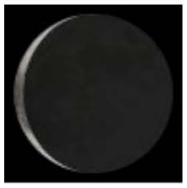
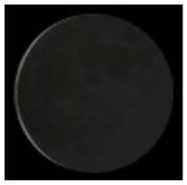
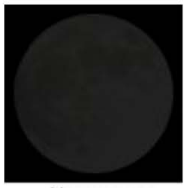
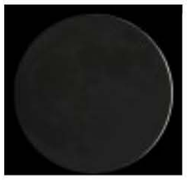
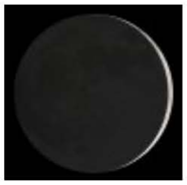
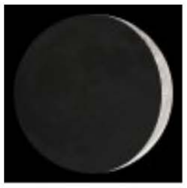
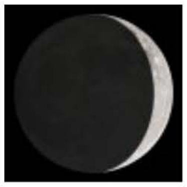
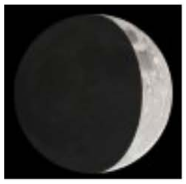
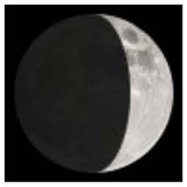
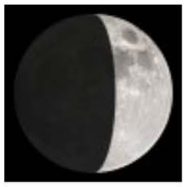
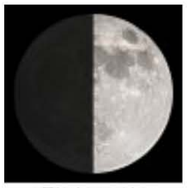
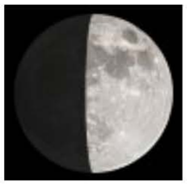
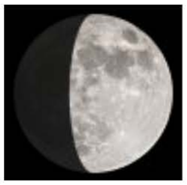
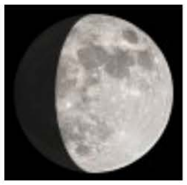
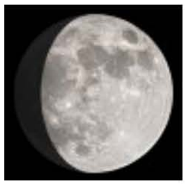
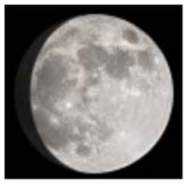
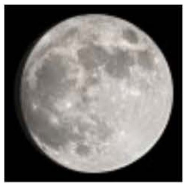
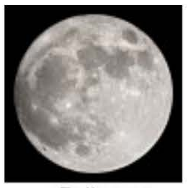
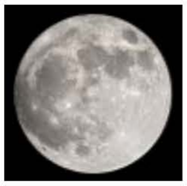
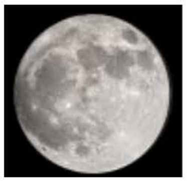
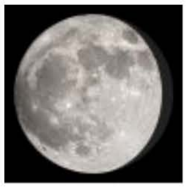
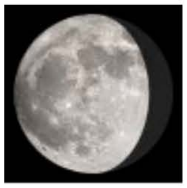
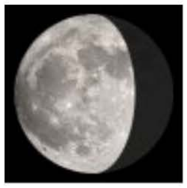
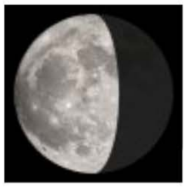
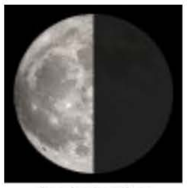
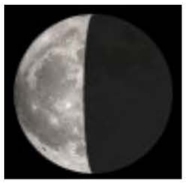
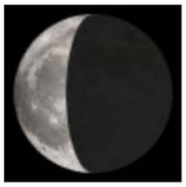
1. Go to scratch.mit.edu and in the search bar enter **et-tech.**
2. Open *"Skeleton Moon Phases"* and change the code for the month of December 2018 — use the December moon calendar provided.
3. Hint: Make sure to change the dates to the correct moon phase and remember the programming concepts we learned.

BONUS: If you have additional time, write a code/script that produces sound. How did you do this? What blocks did you use to accomplish the sound?

Image to determine the phase of the moon.

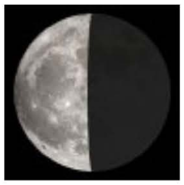
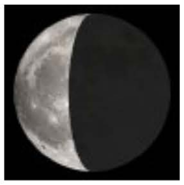
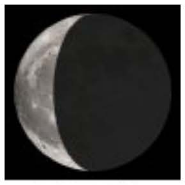
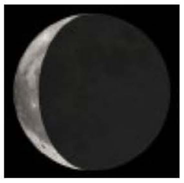
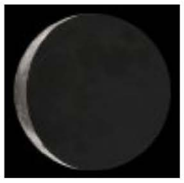
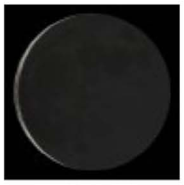
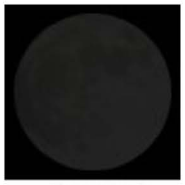
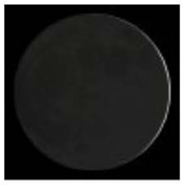
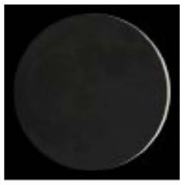
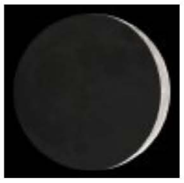
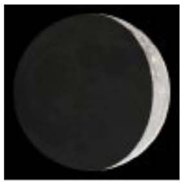
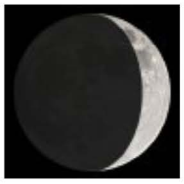
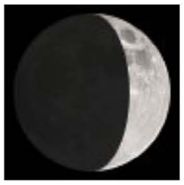
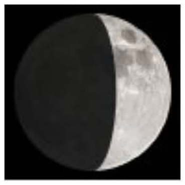
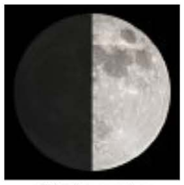
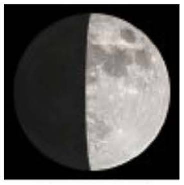
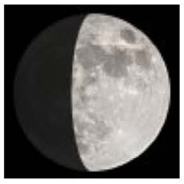
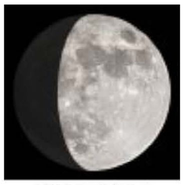
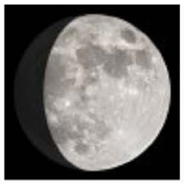
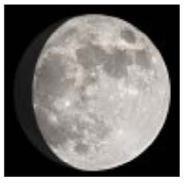
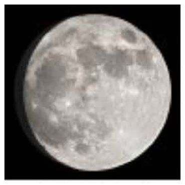
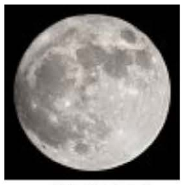
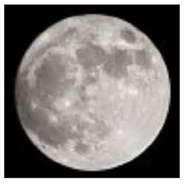
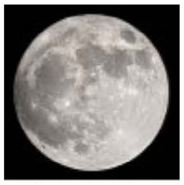
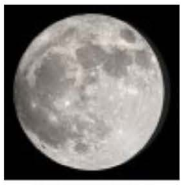
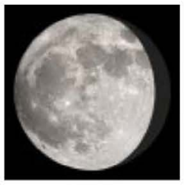
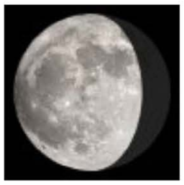
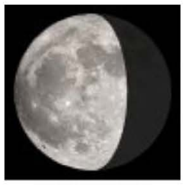
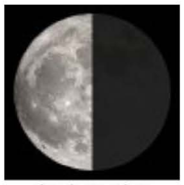
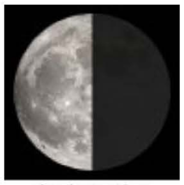


December 2018

Su	Mo	Tu	We	Th	Fr	Sa
						1 
2 	3 	4 	5 	6 	7  New moon	8 
9 	10 	11 	12 	13 	14 	15  First quarter
16 	17 	18 	19 	20 	21 	22  Full moon
23 	24 	25 	26 	27 	28 	29  Last quarter
30 	31 					

https://www.calendar-365.com/moon-calendar/2018/December.html

November 2018

Su	Mo	Tu	We	Th	Fr	Sa
				1 	2 	3 
4 	5 	6 	7  New moon	8 	9 	10 
11 	12 	13 	14 	15  First quarter	16 	17 
18 	19 	20 	21 	22  Full moon	23 	24 
25 	26 	27 	28 	29  Last quarter	30  Last quarter	

https://www.calendar-365.com/moon-calendar/2018/November.html

Name:______________________________

Date:______________ Class:______________

Programmers WANTED
NASA is looking for a contractor to develop a computer program that will help analyze space image data to look at patterns over time. Interested companies must submit a sample computer program and flowchart of a model of visual patterns that change over time meeting the following criteria:

- Events
- One or more loops
- Explain how your computer program works and how it has visual patterns that model change over time.
- Cite evidence on where the events and loop(s) are present that support your change over time and visual patterns model.
- Explain how you created your computer program, the steps you took in designing your program and why that step is necessary in your program.

In addition to your program, you will also explain why you are qualified for the position. In your write-up, include the following:

- An introduction of who you are and the skills that you have to be a computer programmer and how you collaborated with others in developing your program.
- Why you are qualified for the position. Here you might reference your program and how it meets their needs and how you might contribute in analyzing space image data.
- Have a peer write a reference on your behalf, pointing out when you were collaborating.

NASA Submission

[Insert an image, video, or gif of your computer program]

Explain how your computer model works and how it models change over time using patterns.

Using screenshots as evidence, indicate where the events and loop(s) are present in your program and how it supports visual patterns/change over time in your program

[Insert screenshots here]	[Insert screenshots here]
Supports visual patterns/change over time	Supports visual patterns/change over time

Explain how you created your computer program below:

Step	**Why is it necessary in the program?**
1.	
2.	
3.	
4.	
5.	

Personal Introduction
Provide a personal introduction of who you are and the skills you have as a computer programmer. In your introduction, describe why you are qualified for the position and reference how your computer program meets NASA's needs. Also, describe how you can contribute in analyzing space images. Lastly, discuss how you collaborated and were flexible with others in creating your program. You will need to submit at least one reference verifying your collaboration and flexibility with others.

References

Please provide a written reference with examples on how the candidate displayed flexibility and compromise when working in a group to develop the moon phase computer program.

SE RUBRIC: Demonstrating Change Over Time using Moon Phases

	Below Standard	Approaching Standard	Meeting Standard	Above Standard
Computational Thinking skills **3-5.AP.13 Decompose problems into smaller, manageable tasks which may themselves be decomposed**	*Areas that individual students may need one-on-one support with:* • Identifying the overall problem • Breaking the overall problem into at least two tasks • Breaking the two tasks into smaller manageable tasks • Identifying the purpose behind the flowchart shapes and arrows • Effectively using a flowchart to breakdown a problem.	• Flowchart is not sequential • Flowchart is missing a starting point and is comprised of 3-5 Time/Action statements • Flowchart is missing connecting arrows or lacks a logical relationship to the connecting shapes • Flowchart details do not match the program's output *Areas to strengthen might include:* • Identifying smaller manageable tasks • Explaining how to use a flowchart and what the shapes and arrows mean.	• Flowchart is documented in a logical, clear sequence • Flowchart is comprised of a start point, 6 or more Time/ Action statements, and all connecting arrows have a logical relationship to the connecting shapes • Flowchart details match the program's output.	*Areas where student may exceed:* • Thoroughness and detail of the flowchart documentation • Explaining the reasoning behind the chosen shapes and arrows in the flowchart. • Flowchart is comprised of 9 or more Time/Action statements *Ideas for next steps for growth:* • Ask students if there's a more effective and efficient way of representing the program using the flowchart
Coding Skills **3-5.AP.12 Create programs that include events, loops,** *and conditionals.*	*Areas that individual students may need one-on-one support with:* • Understanding the purpose of events and loops in a program • Identifying where in the environment events and loops are located • Identifying how to create a stand-alone event or loop	• Program includes events and loops, but don't function as intended • Uses 1 event and 1 or 0 loops *Areas to strengthen might include:* • Connecting how events and loops help with the creation of a program • How to modify an event or loop in order for it to function properly • Creating stand-alone events and loops that are successful/ function as intended	• Creates program that includes events and loops that function as intended • Uses a minimum of 2 events and 1 loop that function properly throughout the program	*Areas where student may exceed:* • Modifies events and loops in order to enhance the program's overall efficiency *Ideas for next steps for growth:* • Ask students if there are any nested loops and how those can be modified for program efficiency
Collaboration **Exercise flexibility** and willingness to be helpful in **making necessary compromises to accomplish a common goal** (www.p21.org Collaboration2)	*Areas that individual students may need one-on-one support with:* • What flexibility and compromise look like/sound like and what it doesn't look like/sound like • Language/helpful phrases for communicating in a flexible way/ engaging in respectful debate • Strategies for working through conflict/ disagreement	• 1 or very few instances documented of how student was flexible *Areas to strengthen might include:* • Examples given don't really show flexibility or compromise • Examples don't align with teacher observations or other team member reports of team dynamic	• Documented 2 or more instances of how individual student was flexible and helpful in making compromises with other team members.	*Areas where students may exceed:* • Self-reflection shows students notice where they were not flexible/ had trouble being flexible and how they corrected that or plan to improve in the future • Demonstrated ability to help others try something different who might be *Ideas for next steps for growth:* • Make students aware of their strength in this area and how this will benefit them in 21st Century careers. • Encourage them to identify other aspects of collaboration that they struggle with and how they can work on these (e.g. listening) • Increase self-reflection in how their collaboration skills and areas for growth change over multiple projects.