Code Academy
TEACHER'S GUIDE

Computational thinking describes how computer scientists think. They solve problems and achieve tasks by looking for patterns, breaking up problems into smaller steps, prioritizing, and creating step-by-step instructions. The engaging series *Code Academy* uses a narrative style to introduce readers to coding and computational thinking concepts. By reading the stories of a group of school friends, readers will be engaged and interested to learn more. They will also make text to self connections.

The *Code Academy* Teacher's Guide explores computational thinking themes such as logic, if/then statements, and problem solving. Students will also learn about computer safety and encourage their peers to engage in safe computer practices too. Students participating in the lessons in this guide will be encouraged to make text-to-text, text-to-world, and text-to-self connections. They will be encouraged to consider how they use computational thinking and coding strategies in their own lives, and how people use logic and debugging each day.

The lessons in this guide are tailored for grade 2, with an interest level of grades 2 to 4. They can be taught as stand-alone lessons, or sequentially as a unit. Each lesson is inquiry based and encourages students to engage with computational thinking concepts, while making connections to themselves and the real world. The lessons can be done "unplugged" and are accessible across different classrooms. Each lessons is accompanied by reproducible worksheets and assessment tools. The titles in *Code Academy* are:

Code Confusion!
Debugging Disaster!
Hack Attack!
Loopy Logic!
Memory Madness!
Perfect Program!

Ontario
Ontario Media Development Corporation
Made possible with the support of the Ontario Media Development Corporation.

CRABTREE
PUBLISHING COMPANY
WWW.CRABTREEBOOKS.COM

PACING CHART AND VOCABULARY

Lesson Plan Title	Pacing	Vocabulary
Speak My Language: Learning about communication and programming	2 class periods*	code communicate computer instructions language programs programming software
Logic, Processes, and Problem Solving	4 class periods	algorithm bug debug flow chart instructions logic prioritize process statement
Digital Smarts	2 class periods	anti-virus software brochure computer virus hacking password username

* 1 class period = 40-60 minutes

ACCOMMODATION STRATEGIES

Accommodations provide equal access to learning and equal opportunity to demonstrate what is learned. Accommodations allow a student access to the subject or course without any changes to the knowledge and skills the student is expected to demonstrate.

Educators are encouraged to adapt the instructional approach, activities, and assessments included in this guide to best meet the diverse interests, needs, and abilities of their students. Possible accommodations may include:

Instructional Strategies

- Break tasks into parts with accompanying time lines
- Provide extra time for processing of oral information.
- Pair oral instructions with visual ones (writing or symbols)
- Pre-teach new vocabulary and regularly review previously taught vocabulary
- Provided model of completed work
- Frequently check with the student to get him/her started
- Provide oral and visual instructions and examples
- Provide a checklist of tasks for the student

Environmental Strategies

- Proximity to teacher
- Strategic seating
- Flexible or mixed-ability grouping
- Provide an alternative setting for learning that is free from visual and auditory distractions.

Assessment Strategies

- Build in extra time to allow student to process questions asked and answers given
- Provide written instructions and rubrics for assignments
- Offer a choice of assessment activities so that the student can choose one suited to their strengths
- Space out or extend assignments to prevent student feeling overwhelmed
- Reduce the number of tasks used to assess skill or concept
- Allow students to use assistive devices or technology

LESSON 1
Speak My Language: Learning about communication and programming

Curriculum Correlations

Common Core State Standards
CCSS.ELA-LITERACY.RI.2.2
CCSS.ELA-LITERACY.RI.2.6
CCSS.ELA-LITERACY.RI.2.10

Ontario Language Arts
Grade 2 Reading
1.4, 1.6

Materials

- Code Academy books: *Perfect Programming, Memory Madness, Code Confusion*
- White board and markers
- Sticky notes with pencils
- Materials to complete setting the stage "task"
- *My Software Worksheet*
- *My Software Good Copy*
- *My Software Checklist*

Setting the Stage

Spark discussion by asking students: What do we use computers for? Have students give answers either by raising their hands, or using popcorn style. Help students realize that we use computers to make work easier—and to have fun!

Explain to students that today, we will learn about how computers can do work and complete the tasks we need them to.

Hand students sticky notes and make sure they have pencils with them. Then split the class in two groups. One group listens to a read-aloud of *Perfect Programming*, and the other reads *Code Confusion*. While one group is listening to the book read aloud by the teacher, the other group can look through their book and familiarize themselves with it.

Ask students to write on their Post-it note the most important idea in the book they read. Have students stick their notes to the white board and review students' responses. Review some main ideas in each book:

Perfect Programming
- Programs on a computer tell it how to do work
- Programs have instructions that tell a computer what to do
- The programs on computers are called software

Code Confusion
- Computers listen to instructions
- The instructions are called code
- The instructions need to be in a language that the computer can understand

Review how the main ideas in each book connect and overlap. Discuss with the class.

Then, explain to students that computers think and work similarly to how we do! Computers follow code, and we follow instructions.

Walk students through a familiar example of how we follow instructions to complete a familiar task. Choose the task beforehand so that it's possible to prepare the materials to try the steps. Some suggestions include a morning routine, or making a sandwich.

Have students volunteer suggestions for the steps to complete the activity, then write the steps on the white board. Then, go through the steps to complete the activity in front of students. Have students notice if anything is missing or if the steps correctly lead you to complete the task. Make any corrections that are needed to complete the task correctly.

Objectives

Students will:

- Define programming as putting instructions on a computer, and software as written instructions to a computer.
- Show their understanding of coding by communicating messages and instructions using a created coding language, and regular language.
- Create a list of instructions that clearly communicates how to correctly achieve a task.

Debrief by chatting with students about how they needed to change the instructions to make sure the task was completed correctly. Connect to how computers need to follow correct instructions to complete tasks.

Encourage students to make text to self connections by explaining how the robot in each story needed correct or understandable instructions to complete her task.

Extend by asking students to think about another time in their life that they follow instructions to complete a task, or needed to give/improve better instructions to make sure a task was completed correctly.

Activity

Explain to students that they will create a set of instructions to complete their favorite activity. Their instructions ("code") are for a computer, so they need to be clear and have enough specific instruction so that the computer can complete the task correctly. Tell students that the activity needs to be something that anyone can do. Give some examples, such as:

- Choreographing or learning a dance
- Playing a board or card game
- Creating art
- Writing a song or a story
- Learning a magic trick

Hand each student a copy of the *My Software Worksheet*. Each student needs to fill in the worksheet with the steps of the instructions.

When students are finished, they should switch their worksheet with a partner. Their partner should write two or three suggestions for their classmate to improve the clarity of the instructions, similar to the setting the stage activity. Then, give students time to incorporate the suggestions and write out a good copy of their instructions.

Extensions

- Have students read *Memory Madness*. Review the idea that instructions need to be saved on a computer in a pathway. Have students "save" their instructions on a computer by writing their own pathway—how would they make their instructions easy to find? Would they use an external storage system? Why or why not?
- Have students switch instructions and carry out the tasks either at home, or in the classroom after bringing in necessary materials. Have students reflect on how clear the instructions were and whether improvements were needed.
- Have students imagine what might happen if they are given instructions in a language they do not understand. Have them write a story that describes the scenario.
- If students are involved in a coding program, have them build on what they learned about creating instructions by creating a simple code to instruct a computer to complete a task.

Wrap-Up

Encourage students to reflect on the activity by writing a journal entry or completing a reflection worksheet. Have a class discussion and ask students prompting questions such as:

1. Why is it important to write clear, specific instructions?
2. Why is it important to review instructions?
3. How did you improve your instructions to make them work better?

Assessment

Collect the *My Software Worksheet* and *My Software Good Copy* and assess using the *My Software Checklist*. Take anecdotal notes as students participate in the lesson.

Name: _____ **Date:** _____

My Software Worksheet

TASK	

INSTRUCTIONS

1	
2	
3	
4	
5	
6	
7	
8	
9	

10	
11	
12	
13	
14	
15	

Reviewer's Name: _____

IMPROVE THE INSTRUCTIONS BY...

1	
2	
3	

Name: _____ **Date:** _____

My Software Good Copy

TASK	

INSTRUCTIONS

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Student Name: _____ **Date:** _____

My Software Checklist

Criteria	YES / NO
Student created clear instructions to achieve a task or complete an activity.	
Student reviewed a classmate's instructions and made at least two suggestions to improve the clarity of the instructions.	
Student took on the suggestions from classmate to improve their instructions.	
Student created a good copy with minimal errors.	

Additional notes:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

LESSON 2
Processes and Problem Solving

Curriculum Correlations

Common Core State Standards
CCSS.ELA-LITERACY.RI.2.2
CCSS.ELA-LITERACY.RI.2.6
CCSS.ELA-LITERACY.RI.2.10
CCSS.ELA-LITERACY.RI.2.7

Next Generation Science Standards
K-2-ETS1-1.

Ontario Language Arts
Grade 2 Reading
1.4, 1.6

Materials

- Code Academy books: *Debugging Disaster, Loopy Logic*
- Chart paper and markers
- Smart Board (if available)
- *Flow Chart Logic Worksheet*
- *My Algorithm Worksheet*
- *Debug the Algorithm Worksheet*
- *Processes and Problem Solving Exit Card*

Objectives

Students will:

- Define, in their own words, the following terms: logic, process, debug, algorithm.
- Create a flow chart that shows a process of their choice.
- Break down a problem into different steps to create an algorithm.
- Debug an algorithm to show problem-solving skills.

Setting the Stage

Over two different class periods, read and review the books *Debugging Disaster* and *Loopy Logic*. Engage in close reading activities to pique interest and improve comprehension. Suggestions for before, during, and after reading are:

- Before Reading
  - Show students the cover of the book. Ask them what they think the book will be about. Have them raise any questions about the title and pictures.
  - Read through the Table of Contents and have students identify which topics they are most excited to learn about and which topics they might already be familiar with. Assess prior knowledge.
- During Reading
  - When you reach glossary words, stop and have students look at the glossary to find out its meaning. Use clues on the page to find meaning too.
  - Stop after reading a page and ask students to retell the main idea on the page.
- After Reading
  - Retell the main ideas of the book. Make an anchor chart with the main ideas.
  - Together as a class, make text to text connections between the two books, then make text to self and text to real world connections. Write them on sticky notes and make a chart on the white board with the notes in each category.

Review key vocabulary from the books. Then split students into four groups.

Each group gets a piece of chart paper and a marker. One member of the group is the writer. Each group gets a word to define in their own words: logic, process, debug, algorithm.

Each group writes their definition on the chart paper, then puts it on the wall at the front of the classroom. Review each definition together as a class, and talk about how well it defines the word. Make any changes together as a class to create class definitions.

Activity #1: Logic

Read the definitions for logic and process out loud. Explain to students that they will explore these ideas in an activity.

Review some ways that logic connects to our everyday life. Each statement has two processes: true or false. We do something if true, and something different if false. For example:

- Statement: The fire alarm rings
  - True: We follow the fire safety plan
  - False: We stay in our seats
- Statement: The teacher claps their hands
  - True: We clap back to show we are paying attention
  - False: We continue doing our work
- Statement: The bell rings at recess
  - True: We line up to go inside
  - False: We continue playing
- Statement: The leader says "Simon Says"
  - True: We copy the leader's movement
  - False: We do not make any movements

Have students share other ideas of how logic connects to their life. Write all of the examples on the board and filter the examples to make sure that they are strong ones.

Then, have students pick one example that was shared in class. They will each fill out a *Flow Chart Logic Worksheet*, which takes the logic example through two different statements.

Review the flow chart on page 23 of *Loopy Logic* as an example. Conference with students to make sure they understand the task.

When students have completed their worksheets, have them meet in small groups and share their flow charts with each other. Have a full class discussion about what students learned from each other and how the flow charts were the same and different.

Activity #2: Solve and Debug!

In a second period, read the definitions for algorithm and debug out loud. Explain to students that they will explore these ideas in an activity.

Review that an algorithm is a set of instructions that tells a computer what to do. Go through instructions to complete a familiar activity. Hand students the *My Algorithm Worksheet* to help them follow along and support their understanding.

- Example given on the worksheet is a typical classroom task, but task can be changed depending on class.

Review the worksheet together. If possible, show a large version on chart paper or on a Smart Board so that students can view the example together.

Review pages 14 to 16 of *Debugging Disaster* and ensure that all students understand what a bug is, and its effect on algorithms and computer tasks. Then, on the large chart paper or Smart Board example, create a bug by changing an instruction in a way that changes its meaning.

Ask students:

- Where is the bug?
- How did the algorithm change?
- What do you think will happen now?

Review different scenarios, and try the same process with other bugs. Review different types of bugs that can exist in algorithms:

- Missing instructions
- Missing words
- Incorrect instructions
- Spelling mistakes
- Wrong order

Write the types of bugs on the white board or on a piece of chart paper. Then, review page 17 of *Debugging Disaster*. Tell students that they will now create their own "buggy" algorithm. They will then give their algorithm to a peer to "debug".

Students can use strategies to fix the bugs. Ask students if they remember which strategy Frankie used in the book. (She used the rubber duck method: she read the instructions out loud to a rubber duck, which helped her notice the mistake.) Ask students if they can think of any other strategies they have used for problem solving in the past, and might work well for debugging an algorithm. For example:

- Break up the problem into smaller parts
- Prioritize: look for the most important mistakes first
- Work backward from the solution: what do you want to happen, and which steps need to happen for you to get there?

Place students in groups of two or three and hand each group the *Debug the Algorithm Worksheet*. They will write an algorithm with three mistakes. They can use the examples on the board to help them create the bugs. Conference with them as they complete the worksheet to make sure they are on the right track. When students are finished, switch worksheets with another group. They now need to debug the algorithm.

Extensions

- Have students try debugging using a different strategy from the one they used in this activity. Create a personal debugging toolbox on a laminated piece of paper with two to three strategies that they find works well for them.
- Have students create a story or comic strip that tells a story about debugging, similar to the *Debugging Disaster* book, which tells a story about the Code Academy class.
- If students are involved in a coding program, have them build on what they learned about processes and problem solving by coding a process-driven set of instructions, and debug any issues they experience.

Wrap-Up

Review the *Debug the Algorithm Worksheets*.

Share the worksheets by either displaying them on the wall and doing a gallery walk, or passing them around the room. Have students share the strategy they used to debug the algorithm.

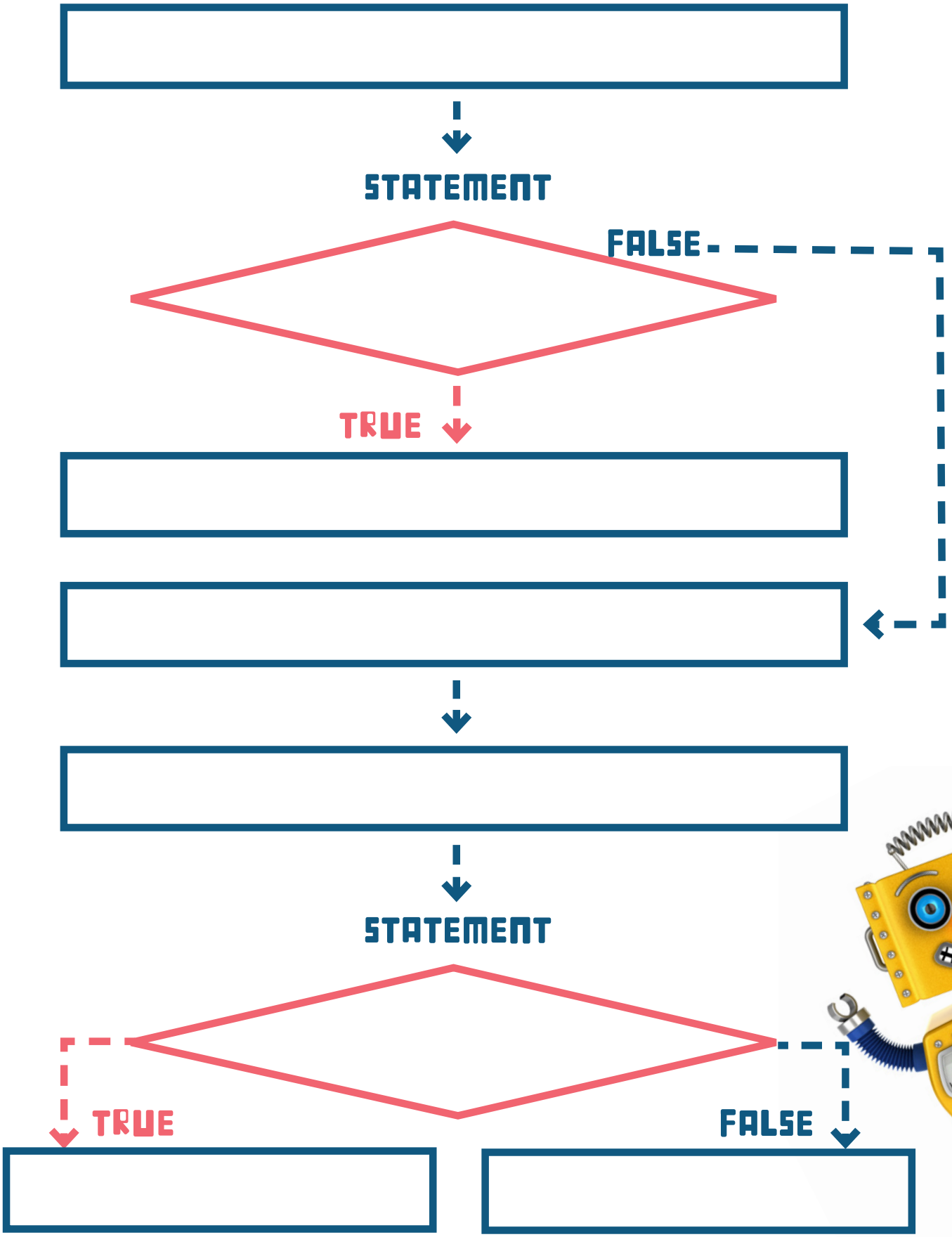
Talk about what worked well, and what they found challenging. Ask students if they think the algorithm they debugged would work as it should, or if it needed more improvement. What would they do differently next time?

Assessment

Have students hand in their *Flow Chart Logic Worksheet, My Algorithm Worksheet*, and *Debug the Algorithm Worksheet*. Assess for completion and understanding. Address any areas of need with students. Collect the *Processes and Problem Solving Exit Card* and use for assessment. Check for student reflection and comprehension.

Name: _____ Date: _____

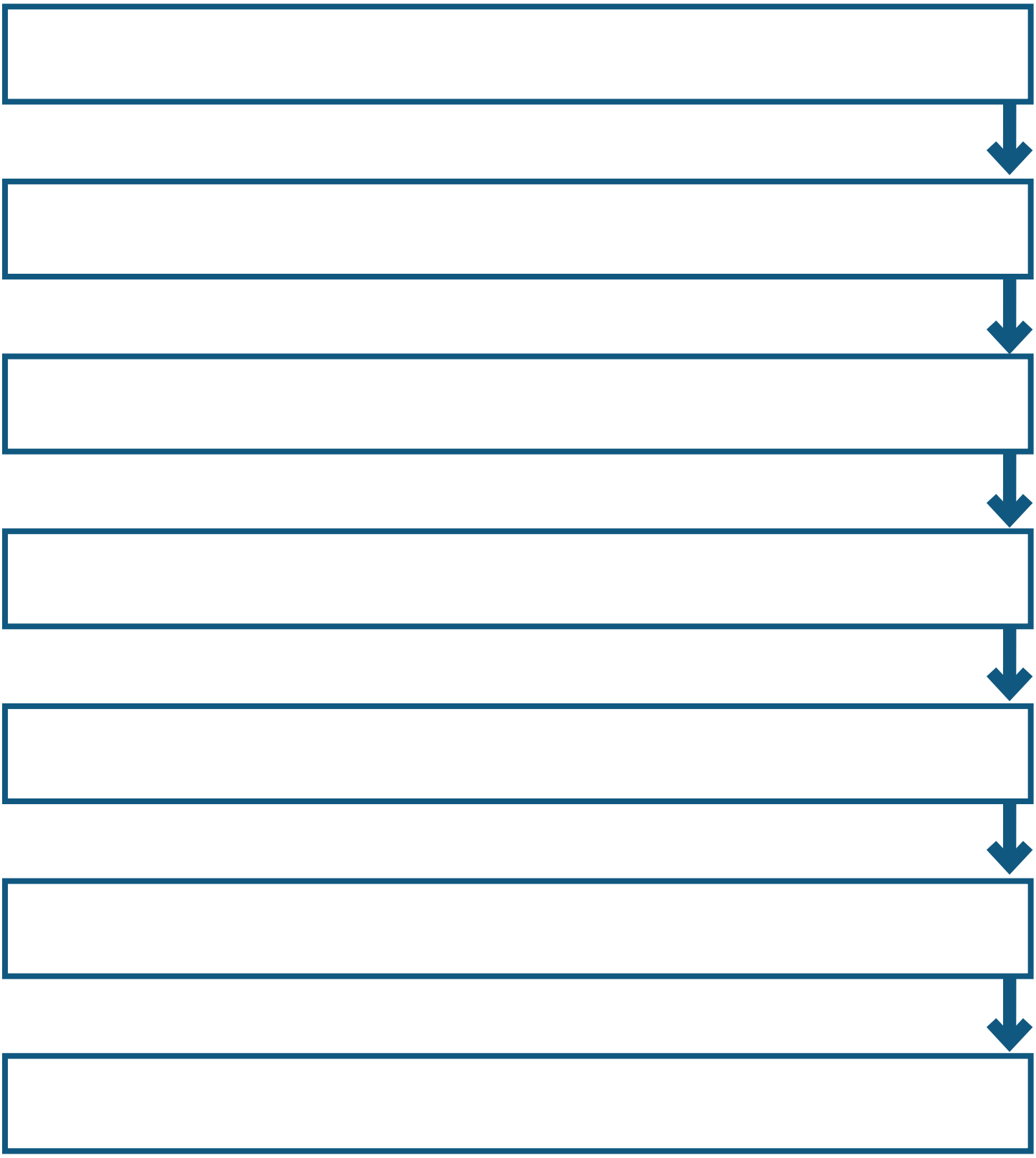
Flow Chart Logic Worksheet



Name: _____ Date: _____

My Algorithm Worksheet

Task | **End of day clean-up**



Name: _____ Date: _____

Debug the Algorithm Worksheet

Task | **End of day clean-up**

Algorithm	Corrections?

Name: _____ **Date:** _____

Processes and Problem Solving Exit Card

1. Give one example of a process you follow each day. /2

_____

_____

2. Describe one way you would try to debug an algorithm. /2

_____

_____

3. Tell me something interesting that you learned, or ask a question about something you would like to learn more about /2

_____

_____

_____

Name: _____ **Date:** _____

Processes and Problem Solving Exit Card

1. Give one example of a process you follow each day. /2

_____

_____

2. Describe one way you would try to debug an algorithm. /2

_____

_____

3. Tell me something interesting that you learned, or ask a question about something you would like to learn more about /2

_____

_____

_____

LESSON 3
Digital Smarts

Curriculum Correlations

Common Core State Standards
CCSS.ELA-LITERACY.RI.2.4
CCSS.ELA-LITERACY.RI.2.1
CCSS.ELA-LITERACY.RI.2.8

Ontario Language Arts
Grade 2 Reading
1.2, 1.3

Grade 2 Writing
1.1, 2.1, 3.1, 3.2, 3.3, 3.4, 3.6

Grade 2 Media Literacy
1.1, 2.1, 3.1, 3.4

Materials

- Code Academy book: *Hack Attack*
- Blank paper and writing and coloring utensils, to create brochure
- White boards and dry-erase markers
- *Computer Safety Brochure Rubric*

Setting the Stage

Read *Hack Attack* to the class. Discuss what students learned by reading the book. Ask discussion questions such as:

- What is hacking?
- What are some ways computer users could be targeted by hackers?
- What are computer viruses?
- What makes a strong password?
- Why are strong passwords important?

Hand each student a white board and marker. Or, give them each a set of cards marked "Safe" and "Unsafe." Then, describe different Internet/computer practices and examples of passwords. Ask students if this practice or password is safe or unsafe. Have students write "safe" or "unsafe" on the white board or draw a thumbs up or thumbs down—whichever they prefer. (Or, they can hold up the cards.)

Some examples of practices or passwords that could be used are:

- Posting your address online (unsafe)
- Using an anonymous username (safe)
- Clicking on a link that was sent to you by someone you don't know (unsafe)
- Keeping your password a secret (safe)
- Use the same password for every website (unsafe)
- Using an anti-virus software on your computer (safe)
- Unsafe password examples:
  - (Name)123
  - (name of city)
  - 12345
  - password
- Safe password examples:
  - PiZza852!
  - cr3@tiv3*
  - 902ScH00l376

Objectives

Students will:

- Identify safe and unsafe computer and Internet practices.
- Create a brochure that uses words and images to educate other kids on computer and Internet safety.

Activity

Explain to students that they will use what they learned to help themselves and others remember and learn computer safety tips. They will make a Computer Safety Brochure that includes three tips. Later, they can use the brochures to make sure they use computers safely!

Hand each student a piece of blank paper and writing and coloring utensils. Show them how to fold the paper along three lines to make the brochure. It might be useful to have a completed example to show students. Review criteria with students. Each brochure must include:

1. A cover page with the words "Computer Safety", name, date, and at least one picture
2. Three computer safety tips—one per interior page, with pictures
3. Conclusion on the back page that answers the question: Why is computer safety important?

Give students a full period to complete their brochure. Instruct them to first complete the brochure in pencil, then proofread the brochure. They should check for spelling and that they wrote complete sentences. Then, go over the pencil in marker or pencil crayon.

Extensions

- Have students present their three tips to the class. Have students reflect on their classmates' presentations and come up with five overall class tips for Internet safety. Share the tips with peers in other classes and review the tips each time students use computers at school.

Wrap-Up

Have students share their brochures by setting them up on desks around the classroom, and walking around to read each other's brochures. Place all of the brochures in a binder or in a display folder and place them in the computer lab, or by the classroom computer. Consult the brochures when needed.

Assessment

Use the *Computer Safety Brochure Rubric* to assess the *Computer Safety Brochure*.

Student Name: _____ **Date:** _____

Computer Safety Brochure Rubric

	Level 1	Level 2	Level 3	Level 4
Knowledge and Understanding	Identifies two or fewer tips for computer safety, and is missing one or both of title page and back cover.	With limited insight, identifies three tips for computer safety and includes title page and back cover.	Identifies three tips for computer safety and includes title page and back cover.	With strong insight, identifies three tips for computer safety and includes creative, engaging title page and back cover.
Thinking	Missing conclusion about why computer safety is important.	Concludes why computer safety is important with limited insight, or conclusion is incomplete.	Concludes why computer safety is important with good insight.	Concludes why computer safety is important with excellent insight.
Communication	Wrote tips using correct spelling and complete sentences, with five or more errors.	Wrote tips using correct spelling and complete sentences, with no more than three or four errors.	Wrote tips using correct spelling and complete sentences, with no more than one or two errors.	Wrote tips using correct spelling and complete sentences, with no errors.
Application	Demonstrated little comprehension of computer safety by including two or less safety tips, with some repetition and unrealistic ideas.	Demonstrated some comprehension of computer safety by including three safety tips, with some repetition and unrealistic ideas.	Demonstrated comprehension of computer safety by including three distinct and useful safety tips.	Demonstrated strong comprehension of computer safety by including three distinct, realistic, creative, and useful safety tips.

Additional notes:

_____

_____

_____

_____

_____

_____

_____

_____

_____