



KAREL JR 2  
STUDENT JOURNAL

REVISED APRIL 19, 2017

NAME	
DATE STARTED	DATE COMPLETED
SCHOOL, CLASS, PERIOD	



## TABLE OF CONTENTS:

WELCOME TO YOUR JOURNAL	4
SECTION 6: IF CONDITIONS	5
SECTION 7: IF/ELSE CONDITIONS; NORTH SENSOR	9
SECTION 8: EMPTY SENSOR; NOT, AND, OR KEYWORDS	13
SECTION 9: WHILE LOOP	17
SECTION 10: COMBINED LOOPS AND CONDITIONS	21
REVIEW YOUR PROGRESS	25
LIST OF BASIC COMMANDS AND KEYWORDS	26
LIST OF KEY VOCABULARY	27
FILE LOG: GAMES I HAVE CREATED	29
DESIGN TEMPLATE	30
NOTES	31

General Website: <https://nclab.com/>

Karel Gallery : <https://nclab.com/karel-gallery/>

Desktop (needs login information) <https://desktop.nclab.com/>

*Keep your name and password in a safe place.*

## WELCOME TO YOUR JOURNAL

**Welcome to Karel 2. By now, you have developed some skills in writing basic commands and repeat loops. You have created a few games yourself.**

As a reminder, this journal empowers you to:

1. Remember better.
2. Create your own reference book.
3. Make connections.
4. Keep a record of your work.
5. Use your notes to collaborate with others.

**This journal is set up the same way as Karel Jr 1.**

The journal is divided into sections that match those in the on-line course. Each section has:

1. One or two **review pages** with questions or activities to help you remember what you have learned. There will always be an open box for your own notes.
2. A **bulletin board** where you can post real life examples: paste in pictures, sticky notes, and scribbles. There are ideas and suggestions at the top of each bulletin board.
3. A **planning page** for your end-of-section project.

The back of the journal contains a **glossary with vocabulary from both Karel Jr 1 and 2**, a **record page** for your files, and a **design template** that can be copied to work on games.

As always, remember to slow down, journal, talk with people, and sketch ideas. We hope you will develop a deeper understanding of what coding is all about, and discover the thrill of having a computer or machine carry out a program that you have written.

Happy coding!

## SECTION 6: IF CONDITIONS

In this section, you learn how to use `if` conditions to check for collectible objects or obstacles, and how to use `if` conditions inside of loops. You also know that the body of conditions is indented the same as the body of loops. Karel can only detect collectible objects which are in his square, and obstacles which are in the adjacent square.

Time for a vocabulary check. Match each term to the correct definition.

TERM	DEFINITION
1. condition	a. a row or column with objects on either side.
2. nested loop	b. a programming line that defines a condition.
3. sensor	c. this tells the program what to look for and how to act.
4. <code>if</code>	d. meet the condition - the condition exists
5. aisle	e. a loop that is within another loop
6. satisfy	f. an item in the program library (in this case Karel), which can be an object, container, or obstacle.

Sensors are used differently depending on what they are.

SENSOR TYPE	ACTION USUALLY TAKEN	IS KAREL IN THE SQUARE OR IN FRONT OF THE SQUARE WHEN HE DETECTS THE SENSOR?	EXAMPLES
Object			
Container			
Obstacle			

**SECTION 6 NOTES**

*Use this space to write your own notes, questions, and problems.*


**QUESTIONS**

A sensor word will change color when typed, if it satisfies two conditions. What are they?

--

Give two examples of how conditions are used in Section 6.

--

Compare conditional loops to `repeat` loops. When would you choose one over the other?

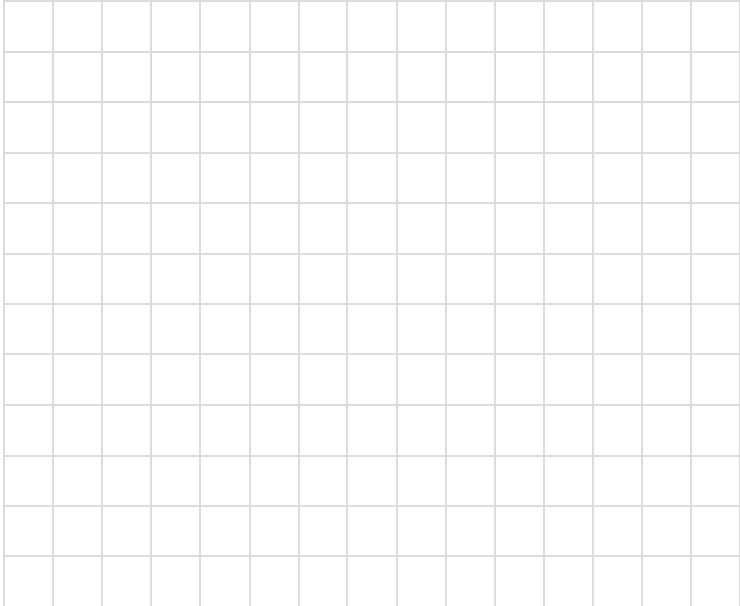
--

## SECTION 6 BULLETIN BOARD

<i>This is a page to post ideas, pictures, sticky notes, drawings</i>	<i>Playing a sport involves many quick decisions.</i>	<i>Think about the conditions and responses involved.</i>

## SECTION 6 PROJECT

Use this page to develop ideas for your game for Karel. The person playing your game will need to write code to solve the game. **Use at least one if condition.** For your notes, you can draw the solution paths through the maze, and write the correct code to the left. **Just make quick notes here: you can make more detailed notes and description in Designer Mode using Edit Game.**

Game Name:		Date:
Program:	Maze Sketch (12 rows x 15 columns)	
		
Storyline:		
Karel's goals:		
Number of steps:	Keywords:	
	Required (must use):	
Number of operations:	Forbidden (can't use):	
Any special challenges:		



## SECTION 7: IF/ELSE CONDITIONS; NORTH SENSOR

In this section, you learn how to use the `else` branch with `if` conditions, and how to use Karel's north sensor. You also know that the body of the `else` branch is indented, the `north` sensor can be used to make Karel point North, and the `north` sensor can be used to make Karel point East, West or South as well. Conditions may contain other conditions or loops, and loops may contain other loops or conditions.

Think of `if/else` as a set of two choices depending on the presence or absence of a sensor. Think of studying for a math test. If you don't understand a type of problem, you will practice it; if you do understand it, you will choose a different type of problem to work on.

Describe two other real life conditions that branch into two choices:


The North Star has been used in navigation for thousands of years. By knowing where North is, we can angle off to any other direction. Karel uses `north` in a similar manner, but he is only allowed to turn 90 degrees at a time by using `left` or `right`. Fill in the table to describe how to end up with East, South, and West by turning right or left. Karel starts out by facing North.

DIRECTION WE WANT KAREL TO FACE	NUMBER OF <b>LEFT</b> TURNS NEEDED	NUMBER OF <b>RIGHT</b> TURNS NEEDED
East		
South		
West		

## SECTION 7 NOTES

*Use this space to write your own notes, questions, and problems.*


## QUESTIONS

Describe two `if/else` conditions from the Section 7 levels. What are the conditions and what are the outcomes? Pick two that you might use yourself when designing a maze.

--

The games are starting to combine different kinds of loops. What would you use to solve the following situations?

Karel follows a wall 14 units long, checking for snakes. When he finds one, he goes around it. If he doesn't, he moves forward.	
---	--

Karel is traveling east. He can't move forward unless he is facing east.	
--	--

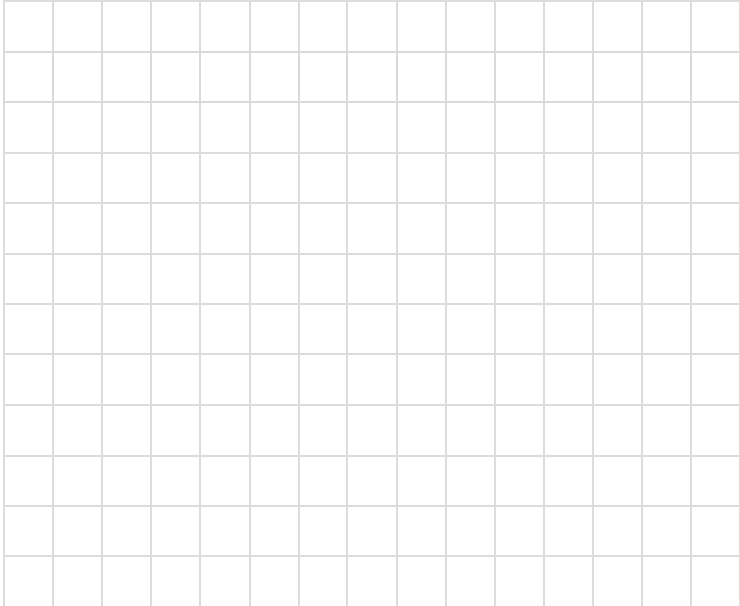
Karel must pick up twenty computer chips. Each time he picks one up, he turns left and goes one step. Otherwise, he goes forward one step.	
--	--

## SECTION 7 BULLETIN BOARD

<i>This is a page to post ideas, pictures, sticky notes, drawings</i>	<i>Imagine an off-roading trip on an ATV or a horse. There are many unmarked trails.</i>	<i>Which ones will you follow? How will you choose? Will direction be a factor? (Think of the north sensor)</i>

## SECTION 7 PROJECT

Use this page to develop ideas for your game for Karel. The person playing your game will need to write code to solve the game. **Use at least one `if/else` condition.** For your notes, you can draw the solution paths through the maze, and write the correct code to the left. **Just make quick notes here: you can make more detailed notes and description in Designer Mode using Edit Game.**

Game Name:		Date:
Program:	Maze Sketch (12 rows x 15 columns)	
		
Storyline:		
Karel's goals:		
Number of steps:	Keywords:	
	Required (must use):	
Number of operations:	Forbidden (can't use):	
Any special challenges:		

## SECTION 8: EMPTY SENSOR; NOT, AND, OR KEYWORDS

In this section, you learn how to use the `empty` sensor to check if Karel's pocket is empty, use keyword `not` to reverse the outcome of conditions, use keyword `and` to make sure that two or more conditions are satisfied at the same time, and use keyword `or` to ensure that at least one of multiple conditions is satisfied. You also know that it is a good idea to use parentheses in more complex logical expressions.

Check your understanding of how these keywords operate by completing the table.

LOGICAL OPERATOR KEYWORD	DESCRIBE THE CONDITION	EXAMPLE
not		
and		
or		

`empty` refers to Karel's pocket, which may contain different amounts of an object. A common real-life problem is, of course, how much money you have in your wallet, or on your bank card, when you go shopping. Think of two other examples of checking a "not empty" condition.

ITEMS IN THE "POCKET"	WHAT HAPPENS IF THE POCKET IS NOT EMPTY? YOU CAN ALSO WRITE AN "ELSE" FOR WHEN IT IS EMPTY.

## SECTION 8 NOTES

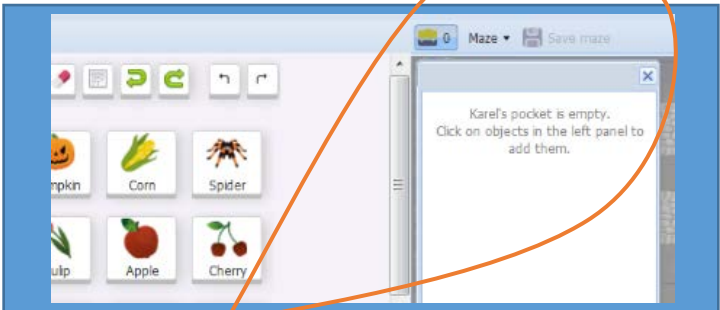
*Use this space to write your own notes, questions, and problems.*

## QUESTIONS

What logical operator or operators would you use for the following conditions? Write out an expression using the operators and parentheses if needed.

Mix chemical A and B together but do not mix them with C. The mixture will explode if you do.	
I can work Wednesday or Thursday next week and Tuesday the following week.	
I will not eat pizza with onions or anchovies.	
Make up your own example using logical operators.	

## SECTION 8 BULLETIN BOARD

<i>This is a page to post ideas, pictures, sticky notes, drawings</i>	<i>Think of quests in real life and in video games. What items do you like to collect?</i>	<i>What purpose do the items serve? How many do you need? Do you need two items to complete a task, or just one or the other?</i>
		
		<p>Have you noticed the pocket counter on the upper left hand corner of the maze? This shows the number of objects in Karel's pocket. He can only put if the pocket contains at least one object.</p> <p>When creating a game, you can pre-load the pocket. In Edit Mode, left-click on the pocket counter and add items. You can load more than one type of item.</p> <p>You may be familiar with bags or other types of storage containers in video games, used to carry supplies, materials, armor, food, and quest items.</p>

## SECTION 8 PROJECT

Use this page to develop ideas for your game for Karel. The person playing your game will need to write code to solve the game. **Use logical operator keywords and try the empty sensor.** For your notes, you can draw the solution paths through the maze, and write the correct code to the left. **Just make quick notes here: you can make more detailed notes and description in Designer Mode using Edit Game.**

Game Name:		Date:
Program:	Maze Sketch (12 rows x 15 columns)	
		
Storyline:		
Karel's goals:		
Number of steps:	Keywords:	
	Required (must use):	
Number of operations:	Forbidden (can't use):	
Any special challenges:		

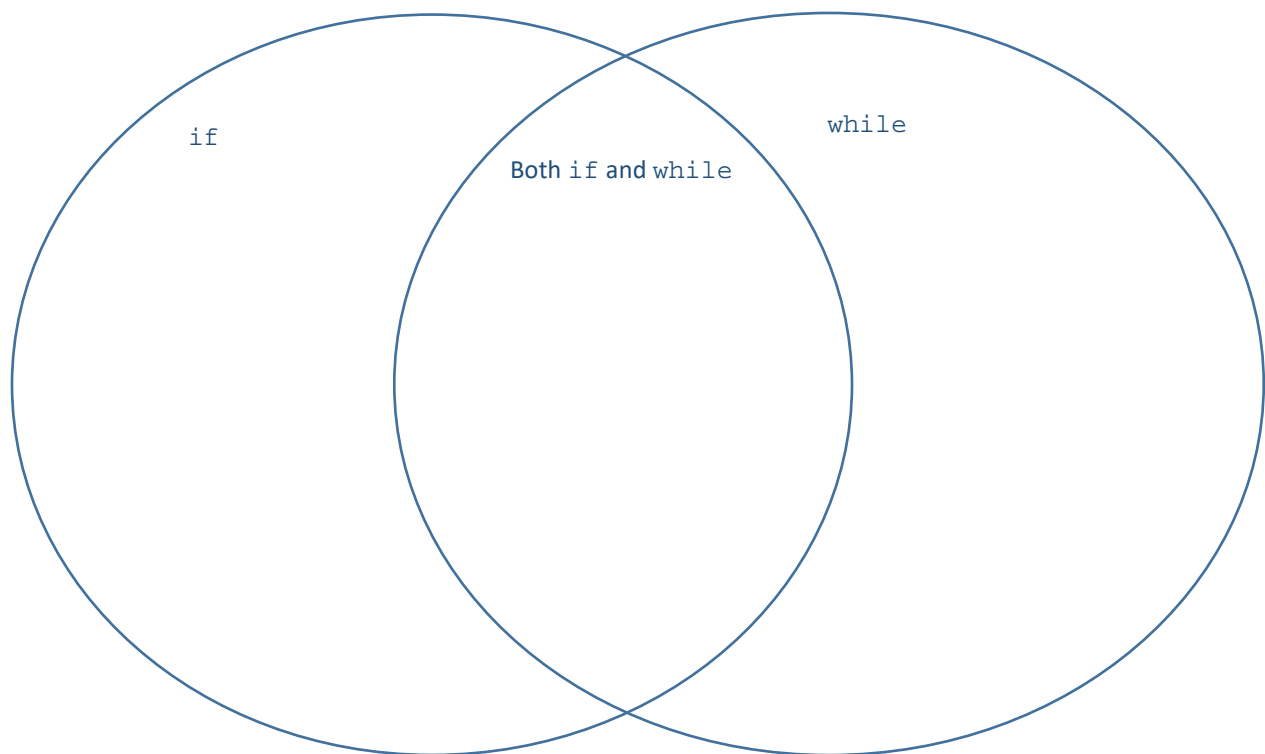


## SECTION 9: WHILE LOOP

In this section, you learn how to use the `while` loop. You also know that the `while` loop is used when the number of repetitions is not known in advance. With `while` loops you can use the same sensors as with `if` conditions. The body of `while` loops is indented same as the body of `repeat` loops.

Let's compare `if` and `while`. They are both conditional, but act differently.

See if you can place these statements in the correct part of the Venn diagram.



Will continue as long as the condition being sensed is present

Use sensors to specify conditions

Will sense each square as a separate test

The number of repetitions is not known in advance.

Can be paired with an else condition

Is a loop

## SECTION 9 NOTES

*Use this space to write your own notes, questions, and problems.*

## QUESTIONS

`while not home` is a commonly used `while` loop. Explain what that means.

Another common loop is `while wall` or the opposite: `while not wall`. What were these loops used for in this Section?

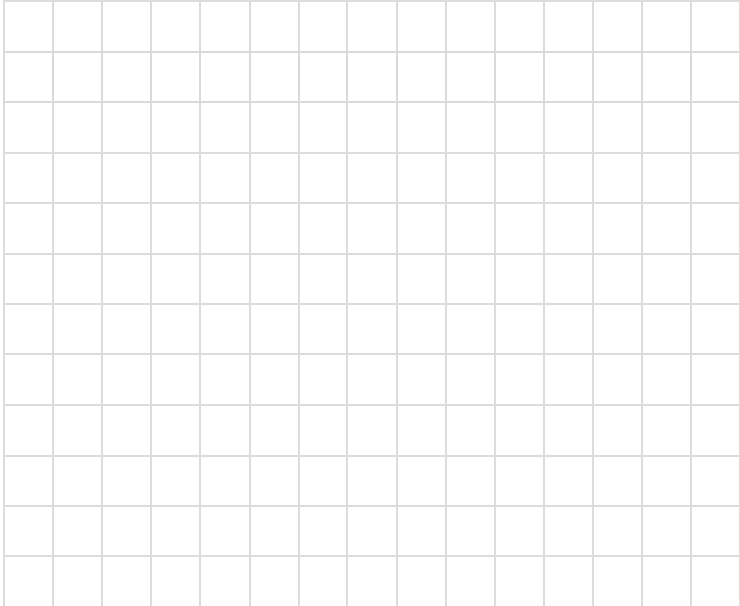
Your SUPER INSPECTOR 9000 is checking a fence for damage. Each board must be examined carefully. The robot won't stop until the whole fence has been checked. What kinds of conditions will you use to program the robot? Will you use `if`, `while`, or both?

## SECTION 9 BULLETIN BOARD

<i>This is a page to post ideas, pictures, sticky notes, drawings</i>	<i>We think we are very good at multi-tasking. While we watch TV, we solve 10 math problems for homework.</i>	<i>Think of other multi-tasking situations.</i>

## SECTION 9 PROJECT

Use this page to develop ideas for your game for Karel. The person playing your game will need to write code to solve the game. **Use at least one** while **loop**. For your notes, you can draw the solution paths through the maze, and write the correct code to the left. **Just make quick notes here: you can make more detailed notes and description in Designer Mode using Edit Game.**

Game Name:		Date:
Program:	Maze Sketch (12 rows x 15 columns)	
		
Storyline:		
Karel's goals:		
Number of steps:	Keywords:	
	Required (must use):	
Number of operations:	Forbidden (can't use):	
Any special challenges:		


## SECTION 10: COMBINED LOOPS AND CONDITIONS

In this section, you learn how to navigate a maze where the path goes either forward, to the left, or to the right. You continue practicing the `while` loop and combine it with other loops and conditions.

There are certain patterns in mazes that are require a specific set of commands. Review 10.1, 10.2, and 10.4 and write down the commands needed to navigate spirals, steps, and the perimeter of a square.

SPIRAL	STEPS (STAIRCASES AND SHELVES)	PERIMETER OF A SQUARE

However, “real world” mazes aren’t so regular. Write down the code explained in 10.6, which will work for navigating any maze.

IRREGULAR AND RANDOM MAZES	
	 <p data-bbox="800 1549 1349 1675">To make copies of your maze, click on the Maze button. Choose “Add random” or “Add a copy”. You can make changes to these copies. Use the colored tabs on the right to select a maze.</p> <p data-bbox="800 1709 1349 1801">Running your program on different mazes is a good way to test that it will run in different configurations without errors.</p>

## SECTION 10 NOTES

*Use this space to write your own notes, questions, and problems.*


## QUESTIONS

Home robots: could they really do our chores? Pick one of these and come up with a plan. Don't try to write the code (it would be very long!), but try to think of tasks that would lend themselves to repeat loops, if conditions, and while loops.

Do the laundry

Clean the bedroom

Shop for groceries

Pull weeds in the garden


Recycle

## SECTION 10 BULLETIN BOARD

<i>This is a page to post ideas, pictures, sticky notes, drawings</i>	<i>You have rated your skills. Make notes to yourself: what have you mastered?</i>	<i>What would you like to learn? What is unclear to you?</i>

## SECTION 10 PROJECT

Use this page to develop ideas for your game for Karel. The person playing your game will need to write code to solve the game. **Make multiple mazes to test your program. Use a variety of commands, sensors, and loops...** For your notes, you can draw the solution paths through the maze, and write the correct code to the left. **Just make quick notes here: you can make more detailed notes and description in Designer Mode using Edit Game.**

Game Name:		Date:
Program:	Maze Sketch (12 rows x 15 columns)	
		
Storyline:		
Karel's goals:		
Number of steps:	Keywords:	
	Required (must use):	
Number of operations:	Forbidden (can't use):	
Any special challenges:		



## REVIEW YOUR PROGRESS

This is the final Section of Karel Jr 2. Reflect on what you have learned so far.

Rate yourself C, B, or A:

- C if you could use this skill any time and could coach someone else;
- B if you have a good understanding but need more practice, and
- A if you feel that you are unsure of yourself and need teaching or coaching.

SKILL OR CONCEPT	C	B	A
if, if/else conditions			
Sensors; logical operator sensors. and, or, not, empty, wall, home			
while loops			
Creating a game in Creative Suite			

Now, set some learning goals based on your self-evaluation. Don't worry if you aren't an expert in everything yet!

<b>RETAKE CERTAIN LEVELS</b> <b>FIND A COACH</b> <b>REVIEW AND DISCUSS NOTES</b> <b>PRACTICE</b>	<b>PRACTICE</b> <b>REVIEW AND DISCUSS NOTES</b> <b>CREATE</b>	<b>READY FOR THE NEXT COURSE</b> <b>CREATE</b> <b>COACH</b>

## LIST OF BASIC COMMANDS AND KEYWORDS FROM KAREL JR 1 AND 2

Command words: `go`, `left`, `right`, `get`, `put`

Directional commands (`go`, `left`, `right`) are always from the robot's point of view.

`go` advances the robot one step.

`left` turns the robot to its left.

`right` turns the robot to its right.

Retrieving and placing objects (`get`, `put`)

`get` picks up an object

`put` places an object

Loops

`repeat x`, where `x` = the number of times the command is to be repeated.

`while x`, where `x` = a defined condition

Conditions

`if x`, where `x` = a defined condition (this may be a single sensor word or a more complex set of conditions using sensor words and operators)

`else` may follow an `if` condition to provide the alternative course of action

Keywords that are Logical Operators

`not` the condition is that the sensor is `not` present

`and` the condition needs all of the sensors joined by `and` to be present

`or` the condition needs one of the sensors joined by `or` to be present

Important Sensor Words (Karel senses objects and containers when he is in the same square.)

`empty` Karel's pocket is empty

`north` Karel is facing `north`, or the top of the maze grid.

`wall` any obstacle is a type of `wall`. Karel senses it in the square in front of him.


`home` the `home` square. Karel sense it when he is in that square.


## LIST OF KEY VOCABULARY FROM KAREL JR 1 AND 2 (IN ORDER OF APPEARANCE)

**Command words:** `go`, `left`, `right`, `get`, `put`. These words tell Karel what to do.

**Home** is the destination square, marked by red diagonal stripes which change to green when Karel approaches the square. The word `home` is also used in conjunction with commands.

**Max** may refer to maximum number of steps, operations, or programming lines.

**Steps** are the number of squares that Karel moves. The shoe icon  counts the number of steps.

**Operations** are anything that Karel does: move, turn, pick up or put down objects. The computer icon  counts the number of operations.

**Objects** are items placed in the maze. (The word “object” can have other connotations in programming that are not used here).

`repeat` is written on its own line as `repeat x`, where `x` = the number of times the command is to be repeated.

**Body:** the body contains the commands to be repeated. The commands are written on the lines following the `repeat` command, indented two spaces.

**Loop:** A set of commands repeated a given number of times.

**Nested loop:** A loop that is within another loop.

This is a good time to introduce some of the terms used in programming. Refer to the online textbook under Section 5 Programming for details.

**Algorithm:** a series of logical steps that leads to the solution of a task. Students may be familiar with algorithms used in operations such as subtraction and long division.

**Logical error:** a mistake in an algorithm. Planning helps reduce the number of errors.

**Computer Program:** An algorithm written using a programming language.

**Syntax:** the way a command line is written.

**Syntax error:** a mistake in spelling, operators, indentations, spaces

**Sensor words:** items from the Karel library, which can include collectible items (such as `orchid`), containers (such as `basket`), and obstacles (such as `wall`, `plant`). A word that is both in the library and correctly spelled will be blue-colored. Collectible and container items

are sensed in the square that Karel occupies. Obstacles are sensed in the square in front of Karel.

**if** is written on its own line as `if x`, where `x` = a defined condition. In these lessons, predefined objects from the library are used as sensor words for the condition.

The body contains the commands to be followed if the `if` condition is met. The commands are written on the lines following the `if` condition, indented two spaces.

**Condition (Section 8 in the textbook):** tells the program what to look for and how to act. Conditions make decisions while the program is running and handle unexpected situations. The program may need to collect all the coins it finds, but may not know where the coins will be located. The `if` condition says: "Is there a coin? If there is a coin, get it." Conditions work like a switch. Note: because `if` conditions test each instance separately, they are NOT loops, even though they are written with a similar format.

**Satisfy:** in programming, satisfy means to meet the condition - the condition exists.

**Aisle:** a row or column with objects on either side

**Sensor:** the presence of something, such as a coin, used to create a condition.

`north`: "`if not north`" can be used to detect if Karel is facing north (the top of the maze), and can be used to reorient Karel to any direction, once he is facing north.

**Key words** `or`, `and`, `not`:

`or`, `and`, `not` are logical operators for the condition. In order to execute the command, `or` means that one (or a set of conditions within parentheses) of two or more conditions must be met, `and` means both or all the conditions must be met, `not` means that condition must not be met.

`empty`: tells whether or not the robot has an object in its pocket. This creates a condition, either `if empty`, or `if not empty`

`while`: A `while` loop is a repeated set of commands that will continue as long as the condition being sensed is present. The number of repetitions is not known in advance. The `while` loop continues until the condition is no longer sensed. `while` loops use the same sensors as `if` conditions. A `while` loop is different because it continues until the condition is no longer sensed, whereas the `if` condition senses each square as a separate test.

**Infinite loop:** If a loop never senses when to end (the stopping condition), it can continue infinitely. Fortunately, most programs will time out if this happens. In Karel, programs can always be stopped manually if this happens.

## FILE LOG: GAMES I HAVE CREATED

FILE NAME AND LOCATION	DATE	DESCRIPTION	NOTES
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			



## NOTES