Karel the Robot – Review

- Primitive Commands
  - move
  - pickbeeper
  - putbeeper
  - turnleft
  - turnoff

- Karel’s program statements are separated by a semicolon (;)
Karel the Robot – Review

Karel’s programs have a standard format.

beginning-of-program

Put the definition of new instruction here

beginning-of-execution

Primitives

Use newly defined instructions

turnoff;

end-of-execution

end-of-program
Karel the Robot – Review

Karel can “learn” new words by using the DEFINE-NEW-INSTRUCTION <new instruction here> AS BEGIN Instructions from primitives or; Instructions from other new instructions listed above this one; END;
For Example:

define-new-instruction Turnright as
begin
  turnleft;
  turnleft;
  turnleft;
end;
Karel the Robot – Review

Karel can repeat instructions by using the Iterate Command.

ITERATE <some number> TIMES
Begin
<instruction(s)>;
End;
For Example: Iterate 5 times
begin
  move;
end;
Karel the Robot – Making Decisions

“Teaching” Karel to Make Decisions

To make decisions, Karel uses the IF statement.

- Basically, Karel can determine if a limited set of conditions is True or False. (pp 66-67 of text)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>front-is-clear</td>
<td>front-is-blocked</td>
</tr>
<tr>
<td>left-is-clear</td>
<td>left-is-blocked</td>
</tr>
<tr>
<td>right-is-clear</td>
<td>right-is-blocked</td>
</tr>
<tr>
<td>next-to-a-beeper</td>
<td>not-next-to-a-beeper</td>
</tr>
<tr>
<td>any-beepers-in-beeper-bag</td>
<td>no-beepers-in-beeper-bag</td>
</tr>
</tbody>
</table>
Karel the Robot – Making Decisions

What does an IF statement look like?

IF <test condition> THEN

BEGIN

<instruction>;

END;

For Example:

If front-is-clear THEN

begin

  move;

end;
**Problem Statement:** Karel is required to jump over a long row of hurdles picking up beepers placed between the hurdles as Karel finds them. The beepers are randomly placed and Karel cannot be sure the beepers will be in the same place the next time Karel goes through the hurdles. Karel will come home and face North.

**Define Output:** Come home after going over the hurdles with all the beepers found after the hurdles in the beeper bag.

**Define Input:** Karel has no beepers in his beeper bag when he starts out. Karel starts out facing North.
Karel the Robot – Making Decisions
Karel the Robot – Making Decisions

Initial Algorithm

- Move to 6th street
- Turn right
- Jump over the hurdle
- Check if beeper
- If beeper pick it up
- Jump over the hurdle
- Check if beeper
- If beeper pick it up
- Jump over the hurdle
- Check if beeper
- If beeper pick it up
- Jump over the hurdle
- ... <keep repeating action>
- After all hurdles are jumped
- Turn right
- Go home
- Turn around

Copyright © 2008 by Helene G. Kershner
Karel the Robot – Making Decisions

Check if beeper
   If next to beeper, pick it up

IF next-to-a-beeper THEN
   BEGIN
      pickbeeper;
   END;
Karel the Robot – Making Decisions

Revised Algorithm

Move to 6th street
Turn right
Jump over the hurdle
If next to beeper, pick it up
Jump over the hurdle
If next to beeper, pick it up
Jump over the hurdle
Jump over the hurdle
If next to beeper, pick it up
Jump over the hurdle
... <keep repeating action>
After all hurdles are jumped
Turn right
Go to 4th street
Go to 1st avenue
Move to Origin (home)
Turn around

Copyright © 2008 by Helene G. Kershner
beginning-of-program
define-new-instruction turnright as
begin
  turnleft;
  turnleft;
  turnleft;
end;

beginning-of-execution
iterate 5 times
begin
  move;
  turnright;
  move;
  if next-to-a-beeper then
  begin
    pickbeeper;
  end;
  turnleft; turnleft;
  move;
  turnright;
  move;
  if next-to-a-beeper then
  begin
    pickbeeper;
  end;
  turnleft; turnleft;
  move;
  turnright;
  move;
  if next-to-a-beeper then
  begin
    pickbeeper;
  end;
  turnleft; turnleft;
  move;
  turnright;
  move;
  if next-to-a-beeper then
  begin
    pickbeeper;
  end;
  turnleft; turnleft;
  move;
  turnright;
end;

REPEAT A BUNCH
BRING KAREL HOME
Karel the Robot – Making Decisions

beginning-of-program
define-new-instruction turnright as begin
  turnleft;
  turnleft;
  turnleft;
end;
beginning-of-execution
iterate 5 times
begin
  move;
end;
turnright;
move;
move;
turnright;
if next-to-a-beeper then
begin
  pickbeeper;
end;
turnleft; turnleft;
move;
turnright;
move;
turnright;
move;
if next-to-a-beeper then
begin
  pickbeeper;
end;
turnleft; turnleft;
move;
turnright;
move;
if next-to-a-beeper then
begin
  pickbeeper;
end;
turnleft; turnleft;
move;
turnright;
move;
REPEAT A BUNCH
BRING KAREL HOME
Karel the Robot – Making Decisions

beginning-of-program
define-new-instruction turnright as
begin
  turnleft;
  turnleft;
  turnleft;
end;

define-new-instruction turnaround as
begin
  turnleft;
  turnleft;
end;

beginning-of-execution
iterate 5 times
begin
  move;
end;

iterate 12 times
begin
  move;
  turnright;
  move;
  if next-to-a-beeper then
    begin
      pickbeeper;
    end;
  turnaround;
  move;
  turnright;
end;

BRING KAREL HOME
turnoff;
end-of-execution
end-of-execution
end-of-program
**Problem Statement:** Karel starts at the Origin and travels through a simple handed maze turning right at each corner of the maze. Karel returns back to the Origin facing North.

**Define Output:** Karel ends at the Origin facing North.

**Define Input:** Karel starts at the Origin facing North, the maze is above and to his right.
Karel the Robot – Making Decisions

Looking at the maze we know:

1. Each side is 5 spaces long  
This is a good place to use the Iterate Command.

2. Each turn is to the right  
Karel can check to see if he is in the corner.  
This is a good place for the If – Then Command.

3. Everything is repeated 4 times  
This is another good place for the Iterate Command.
Karel the Robot – Making Decisions

Initial Algorithm

- move Karel to entrance of maze
- go up first side
- at corner turn right
- go up second side
- at corner turn right
- go up third side
- at corner turn right
- go up fourth side
- at opening turn left
- move home
- turn right
beginning-of-program
  define-new-instruction turnright as
  begin
    turnleft;
    turnleft;
    turnleft;
    turnleft;
  end;
define-new-instruction move5 as
begin
  iterate 5 times
  begin
    move;
  end;
end;
beginning-of-execution
  turnright;
  move;
  move5;
  turnright;
  move5;
  turnright;
  move5;
  turnright;
  move5;
  turnright;
  move5;
  turnleft;
  move;
  turnright;
  move;
  move;
  turnright;
  turnoff;
end-of-execution
end-of-program
What instruction can we use in our program that will require that Karel check whether a move is possible?

- Look at the Maze, regardless of its size when Karel reaches the corner, Karel makes a right turn.
- But how does Karel know if Karel is at the corner?

An If statement = Decision

What question, using an IF can Karel ask to determine whether a corner has been reached?
Remember: To make decisions, Karel uses the IF statement.

\[
\text{IF } <\text{test condition}> \text{ THEN} \\
\quad \text{BEGIN} \\
\quad \quad <\text{instruction}>; \\
\quad \text{END};
\]

These are the <test conditions>:

- front-is-clear
- left-is-clear
- right-is-clear
- next-to-a-beeper
- any-beepers-in-beeper-bag
- front-is-blocked
- left-is-blocked
- right-is-blocked
- not-next-to-a-beeper
- no-beepers-in-beeper-bag
So, in our example, what question, what test condition can we use to have Karel decide if Karel is at a corner?

- front-is-clear
- front-is-blocked
- left-is-clear
- left-is-blocked
- right-is-clear
- right-is-blocked
- next-to-a-beeper
- not-next-to-a-beeper
- any-beepers-in-beeper-bag
- no-beepers-in-beeper-bag
Karel the Robot – Making Decisions

You can write your program using any of the following:

- front-is-clear
- front-is-blocked
- left-is-clear
- left-is-blocked
- right-is-clear
- right-is-blocked
- next-to-a-beeper
- not-next-to-a-beeper
- any-beepers-in-beeper-bag
- no-beepers-in-beeper-bag
Karel the Robot – Making Decisions

You can write your program using any of the following:
front-is-clear, front-is-blocked, left-is-clear, left-is-blocked,
right-is-clear, right-is-blocked

Depending on which one you choose, your IF statement will be
different and some of your ending instructions might be
different but the basic idea for the program will be the same.

If front-is-clear then
  begin
    move;
  end;
  turnright;
If front-is-blocked then
  begin
    turnright;
  end;
  move;
Karel the Robot – Making Decisions

beginning-of-program
define-new-instruction
turnright as
begin
  turnleft;
  turnleft;
  turnleft;
end;
beginning-of-execution
turnright;
move;
move;
turnleft;
move;
iterate 4 times
begin
  if front-is-blocked then
    begin
      turnright;
    end;
  iterate 5 times
    begin
      move;
    end;
end;
iterate 4 times
begin
  if front-is-blocked then
    begin
      turnright;
    end;
iterate 5 times
begin
  move;
end;
end;
turnleft;
move;
turnleft;
move;
turnright;
move;
move;
turnright;
turnoff;
end-of-execution
end-of-execution
end-of-program
end-of-program
Karel the Robot – Making Decisions

- What if we know that Karel is working on a 4-sided maze where Karel will always need to turn right, BUT we do NOT know in advance how long each side is?
- Our program work only and exactly when the maze is five blocks in each direction.
- How can we design our program (code) so that it works for any right-handed, 4-sided maze?
- The goal in writing a program is to make it general enough to solve any similar problem.
Karel the Robot – Making Decisions

3-squares/wide

5-squares/wide

7-squares/wide
Karel the Robot – Making Decisions

- We could use the same program rewriting it each time to reflect the different number of squares per side on each maze.
- BUT, Doing that would mean every time we have a different maze we would need a different program and the iterate command that moves Karel along the side would change.

```plaintext
iterate 4 times
    begin
        if front-is-blocked then
            begin
                turnright;
            end;
        iterate 5 times
            begin
                move;
            end;
    end;
```
While this was easy to do, it does not give us code that works for any right turning maze.

We, the programmer would need to write, essentially, a new program for each new maze.

BUT, we CAN design Karel programs so that Karel CAN do all the work of deciding when to turn without our having to know how long each side is.

Karel is a computer.

Karel does not get bored.

Karel does not get tired asking the same question over and over and over and over and over and..... again
Let’s examine how Karel asks more complex questions. We will return to our simple Maze problem in a moment.
Problem Statement:
Karel has been told to place two beepers on each street corner between 1st street and 2nd avenue and 1st street and 10th avenue and return home. This would be a very easy task except there was a beeper party last night and there are beepers scattered around on random street corners.

Define Output: There will be two beepers on every corner on 1st street between 2nd avenue and 10th avenue. Karel will be at the origin.

Define Input: Karel is next to 20 beepers. He is at the Origin facing east.
Karel the Robot – Making Decisions

Copyright © 2008 by Helene G. Kershner
Karel the Robot – Making Decisions

Initial Algorithm

Pickup 18 beepers
Move
If next to a beeper, put a beeper down
Otherwise put down 2 beepers
Move
If next to a beeper, put a beeper down
Otherwise put down 2 beepers
Move
If next to a beeper, put a beeper down
Otherwise put down 2 beepers

...  

turnaround
move 9 streets

turnoff

Copyright © 2008 by Helene G. Kershner
Karel the Robot – Making Decisions

Revised Algorithm

Pickup 18 beepers
Repeat 9 times
  Move
    If next to a beeper, put a beeper down
    Otherwise put down 2 beepers
  turnaround
  move 9 streets
  turnoff
Karel the Robot – Making Decisions

**IF / THEN / ELSE**

IF < test condition> THEN
BEGIN
  <Do these instructions when test = True>;
END
ELSE
BEGIN
  <Do these instructions when test = False>;
END;
Looking at our revised algorithm, we have the statement:

If next to a beeper, put a beeper down

Otherwise put down 2 beepers

Anytime we use a word like OTHERWISE in English an IF/THEN/ELSE is used in Karel’s world.

Translating our algorithm into Karel’s language would produce:

IF next-to-a-beeper THEN
BEGIN
    putbeeper;
END
ELSE (another word for Otherwise)
BEGIN
    putbeeper;
    putbeeper;
END;
Karel the Robot – Making Decisions

Looking at Karel’s code

IF next-to-a-beeper THEN
BEGIN
putbeeper;
END
ELSE (another word for Otherwise)
BEGIN
putbeeper;
putbeeper;
END;

1) When Karen is next-to-a-beeper (the test is TRUE), one beeper get put down.
2) When Karen is NOT next-to-a-beeper (the test is FALSE), two beepers get put down.
3) BEGIN / END statements must match up or Karel (not to mention the programmer) gets very confused.

Copyright © 2008 by Helene G. Kershner
Karel the Robot – Making Decisions

beginning-of-execution
ITERATE 18 TIMES
BEGIN
  pickbeeper;
END;
ITERATE 9 TIMES
BEGIN
  move;
  IF next-to-a-beeper THEN
    BEGIN
      putbeeper;
      END
  ELSE
    BEGIN
      putbeeper;
      putbeeper;
    END
  END;
turnaround;
ITERATE 9 TIMES
BEGIN
  move;
END;
turnoff;
end-of-execution

HINT: This code is similar to what is needed to solve your 2nd Project.