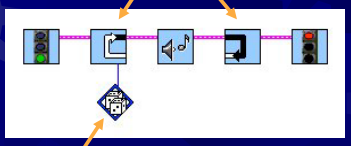
Robots Week 3a
Programming Lesson 2: Modifiers, Loops and Jumps
Programming Solutions

Exercise 1: Solution

Play a beep for a random number of times.

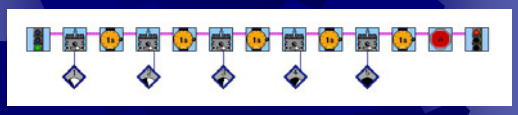
Put the Play Sound icon in a loop



The dice modifier gives a random number

Exercise 2: Solution

Turn on motor A in the forward direction at power level 1 for 1 second, then on power level 2 for 1 second, etc. Continue this pattern up to power level 5 for 1 second, then stop the motor.

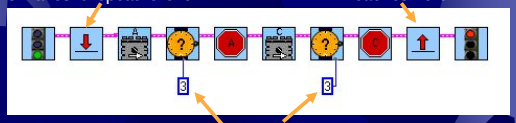


Exercise 3: Solution

Turn on motor A in the forward direction for 3 seconds, then stop the motor. Then turn on motor C in the forward direction for 3 seconds, then stop the motor. Repeat this forever.

...it jumps back to here. This makes it repeat forever.

When the program reaches here...

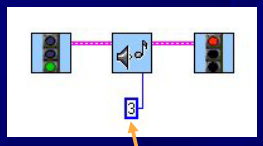


To get 3 seconds, you need a numeric constant modifier

Lesson 2: Modifiers, Loops and Jumps
Troubleshooting Tips

Problem 2a

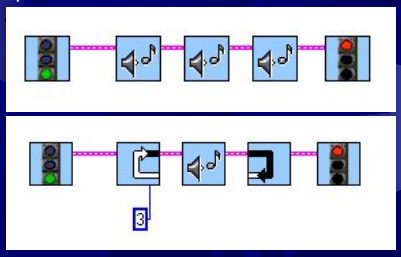
Write a program that plays a beep three times in a row.



This plays sound #3, NOT 3 sounds.

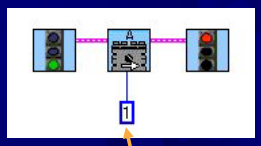
Solution(s) 2a

Use 3 Play Sound icons, or use a loop that loops 3 times.



Problem 2b

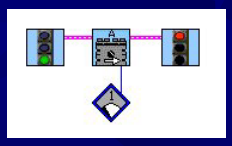
Turn on motor A at power level 1.



This is a numeric constant modifier NOT a power level modifier. (Strangely enough, it does not show a bad wire. Hmmmm…..)

Solution 2b

Use a Power Level 1 modifier.



Problem 2c

Write a program that beeps forever. What's wrong?

But this is a red land

This is a yellow jump

Solution 2c

Make sure jump and land pairs are the same color.

Problem 2d

Write a program that beeps forever. What's wrong now?

Start

Jump to green land

Land here

End

Oops!

Solution 2d

Be careful not to mix up the order of your jumps and lands!

Jumps

- ***Jumps*** are like GoTo commands in other languages.
- Program "jumps" on up arrow and "lands" on down arrow.
- Can create a loop. But it makes an *infinite* loop, which goes on forever.
- It is better to use the ***loop*** command for that and to use ***jumps*** with forks.

Loops

- ***Loops*** are conditionals ("if" statements).
  - They allow you to repeat sections of your program over and over.
  - All start with a ***start loop*** command.
  - All terminate with an ***end loop*** command.
  - Can specify the number of times to repeat.
  - Default is to loop twice.
  - Can nest loops.

Forks

- ***Forks*** are equivalent to "if-then-else" in other languages.
- All fork commands require a ***merge fork*** be used later. (like end if)
- All use a greater than/less than condition.
  - You must wire a threshold value.
  - Touch sensors do not require a threshold, they are just pushed in or not.

Equal forks

- Icon on the far right of the forks sub-menu.
- ***Equal forks*** are the same as standard forks but the conditional statement is "equal to/not equal to" instead of "greaterthan/less than"

Tips

- Displaying data to the LCD panel is a common way of debugging a program.