The following instructions will take you through the steps of creating a game where the computer plays a sequence of keypresses that the player memorises and then plays back.

The first sequence is 4 keypresses long.

Each time the sequence is played back correctly; the length of sequence increases by 1.

There are 3 attempts at each sequence.

The game ends when the player fails to get the sequence correct 3 times.



#### Controls

To control the tank, use the left and right direction keys on the console. Alternatively, it may be easier to use the keyboard mappings for the keys as follows. Use the A button to serve the ball.

	Up			
	(个 or W)			
Left		Right	В	А
(← or A)		$(\rightarrow$ or D)	(E, X or Enter)	(Q, Z or Space)
	Down			
	(↓ or S)			

#### Step 1 – Draw the icons

Because the game generates sequences of keypresses to memorise, the game requires 13 images to be created in the Assets section of MakeCode Arcade. The Assets section is accessed from the buttons on the top bar. Each image should be 16 x 16 pixels.

There will need to be 6 images representing the enabled keys and named "left", "right", "up", "down", "b", and "a".

There will need to be 6 images representing disabled keys and named "left-inactive", "right-inactive", "up-inactive", "down-inactive", "b-inactive", and "a-inactive".

There will need to be 1 image representing a cross and named "cross".

Examples of all 13 images are shown in the screen shot below.



### Step 2 – Setup the top row of icons

The top row of icons will be used as the focus of the game. We will use a function to setup the array of icons and position them correctly. Create a new function called setuplcons by selecting Advanced -> Functions -> Make a Function... as shown below.

<ul> <li>Variables</li> <li>Math</li> <li>Extensions</li> <li>Advanced</li> <li>Animation</li> <li>Images</li> <li>fx Functions</li> <li>Arrays</li> <li>Text</li> </ul>		ƒ(x) Function Make a Funct	ns Lon				0	
Add a parameter	중 Text	≫\$ Boolean	🖬 Number	i⊟ Array	🖌 Sprite	im Image	v	
			function set	tupIcons				
							Done 🗸	
function setup set mySprite	Icons 🔊	orite	of kind Pla	yer •		function set set mySpri mySprite New variable Rename variat Delete the "my	upIcons (a) te v to (a) ble Sprite" variable	

function setupIcons	function setupIcons
set icons ▼ to empty array 🕀	set icons ▼ to array of ⊖ ⊙
function setupIcons	
set icons ▼ to array of sprite	of kind Player 🔻 💬 🕣
function setupIcons	
set icons • to array of sprite	of kind Player 🔻 💬 🟵
function setupIcons	
аггау оf	





#### Step 3 – Setup the sounds to play

This is like the previous step but instead of icons, we are setting up sounds to play. Start by creating a new function called "setupSounds".







#### Step 4 – Setting up the sprites to show

As in the previous 2 steps, a new function is to be created to setup the sprites that will be needed.





### Step 5 – Setup the error indicator

When the player makes a mistake, we will want to let them know y showing a cross. Add the following code to your "on start" block. Start by making a new variable



 $\mathbf{e}$ 



showing

on st	art
cal	l setupIcons
cal	L setupSounds
cal	l setupSprites
set	showing - to false -
set	cross • to sprite X of kind Player •
set set	cross • to sprite of kind Player • cross • invisible • ON
set set	cross • to sprite of kind Player • cross • invisible • ON volume 255
set set	cross • to sprite of kind Player • cross • invisible • ON volume 255

### Step 6 – Dropping the sprites

Here a new function is to be created that will be used to make the sprites drop from the icons.







function dropSprite num 📀
set mySprite 🔹 to sprite sprites 🔹 get value a num of kind Player 🔹
set mySprite → position to x icons → get value at num x → y icons → get value at num y →
set mySprite ▼ velocity to vx 0 vy 50
set mySprite V lifespan V to 750

### Step 7 – Showing the sprites

As in the previous step, create a new function but this time call it show.



for e	Lement value	of list •			
do ca	11 dropSprite	value •			
p	ay tone sou	nds ♥ pet value a	t value 🔻 for	1 ▼ beat u	ntil don
D	use 750 • m	s			

#### Step 8 – Creating a level

Create a new function called NextLevel













### function nextLevel 🐼 set life to 3 set list 🕶 to empty array 🕀 set guesses ▼ to empty array 🕀 repeat score times list 🔹 add value pick random 0 to 10 to end join "Hello show long tex World bottom 💌 call show function nextLevel 🚫 set life to 3 set list ▼ to empty array 🕂 guesses 🔻 to 🛛 empty array 🕂 repeat score times to end list 🔹 add value pick random 0 5 Round 0 0 show long text joi top call show function nextLevel 🚫 set life to 3 set list ▼ to empty array 🕂 set guesses 🔻 to empty array 🕀 repeat score times do list • add value pick random 0 to 5 to end show long text join ("Round" score top 💌 3 call show

#### Step 9 – Hooking up the start

Add the following to the "on start" code; then try it out.



#### Step 10 – Guesses

Here a new function is to be created that will be used to make the sprites drop from the icons.





<pre>if um + ist * get value at 0 then   guesses * aid value nm to end   call dropSprite nm   play tone list * get value a nm for 1 * beat until done *     if length of array list * p length of array list * then     play sound ba ding * until done *     ese</pre>	function guess num 📀
<pre>guesses * add value num to end call dropSprite num play tone list * get value a num for 1 * best until done * if length of array list * i * length of array list * then play sound ba ding * until done * change score by 1 call nextLevel * * * * * * * * * * * * * * * * * * *</pre>	if num = v list v get value at 0 then
<pre>call dropSprite nm play tone list • get value a num for 1 • beat until done • if length of array list • 1 • length of array list • then play sound ba ding • until done • change score by 1 call nextLevel • function guess • und of the set until done •</pre>	guesses  add value num to end
<pre>play tone list * get value a num for 1 * beat until done * if length of array list * 1 * length of array list * then play sound ba ding * until done * change score by 1 call nextLevel * * * * * * * * * * * * * * * * * * *</pre>	call dropSprite num
<pre>isc isc isc isc isc isc isc isc isc isc</pre>	play tone list ▼ get value a num for 1 ▼ beat until done ▼
<pre>int length of array list + let length of array (list + then</pre>	
<pre>play sound ba ding * until done * change score by 1 call nextLevel else function guess num @ function guess num @ function guess add value num to end call dropSprite num play tone list * get value at num for 1 * beat until done * if length of array guesses * 2 * length of array list * then play sound ba ding * until done * change score by 1 call nextLevel </pre>	If length of array list • 2 • length of array list • then
<pre>change score by 1 call nextLevel cale cale cale cale cale cale cale ca</pre>	play sound ba ding  until done
<pre> function guess num o  function guesse num o</pre>	call next evel
<pre>else</pre>	
<pre> Function guess num</pre>	else
<pre>function guess num ② if num = list * get value a length of array guesses * then     guesses * add value num to end     guesses * add value num to end     guesses * add value num for 1 * beat until done *     if length of array guesses * 2 * length of array list * then     play sound ba ding * until done *     change score by 1     call nextLevel </pre>	
<pre>function guess num () if num = list , get value a length of array guesses , then     guesses , add value num to end     guesses , add value num to end     call dropSprite num     play tone list , get value at num for 1 + beat until done +     if length of array guesses , 2 length of array list , then     play sound ba ding , until done +     change score by 1     call nextLevel     e&gt;</pre>	
<pre>function guess num ③ if num = list v get value a length of array guesses v then     guesses v add value num to end     call dropSprite num     play tone list v get value at num for 1 v beat until done v     if length of array guesses v ≥ v length of array list v then     play sound ba ding v until done v     change score by 1     call nextLevel </pre>	
<pre>if num = v list v get value a length of array guesses v then     guesses v add value num to end     call dropSprite num     play tone list v get value at num for 1 v beat until done v     if length of array guesses v 2 length of array list v then     play sound ba ding v until done v     change score by 1     call nextLevel     else</pre>	function guess num
<pre>guesses • add value num to end call dropSprite num play tone list • get value at num for 1 • beat until done • if length of array guesses • 2 • length of array list • then play sound ba ding • until done • change score by 1 call nextLevel</pre>	if list ▼ get value a length of array guesses ▼ then
call dropSprite num play tone list • get value at num for 1 • beat until done • if length of array guesses • 2 • length of array list • then play sound ba ding • until done • change score by 1 call nextLevel	guesses  add value num to end
<pre>play tone list • get value at num for 1 • beat until done • if length of array guesses • 2 • length of array list • then play sound ba ding • until done • change score by 1 call nextLevel </pre>	call dropSprite num
if length of array guesses • ≥ • length of array list • then play sound ba ding • until done • change score by 1 call nextLevel	play tone list • get value at num for 1 • beat until done •
<pre>play sound ba ding • until done • change score by 1 call nextLevel • </pre>	if length of array guesses ▼ ≥ ▼ length of array list ▼ then
change score by 1 call nextLevel	play sound ba ding  until done
call nextLevel	change score by 1
else	call nextLevel

function guess num	
▲ if num = • list • get value at length of array guesses •	then
▲ guesses ▼ add value num to end	
call dropSprite num	
play tone list • get value at num for 1 • beat until done •	
▲ if length of array guesses ♥ ≥ ♥ length of array list ♥	then
pli / sound power up • until done •	
change score by 1	
call nextLevel	
▲ set showing - to 0	
change life by -1	
set cross V auto destroy V OFF	
if true 💌 then	
play sound ba ding    until done	
set guesses V to 0	
pause 100 v ms	
set cross • auto destroy • OFF	
call show	
	function guess num
	At num = 115t v get value at length of array guesses v then
	guesses  add value num to end
	call dropSprite num
	play tone list • get value at num for 1 • beat until done •
	if length of array guesses • 2 • length of array list • then
	play sound power up vi until done vi
	Change score by 1
	else 🔘
	set showing - to true -
	change life by -1
	set cross • invisible • OFF
	if 0 < 0 then
	play sound ba ding - until done -
	set guesses ▼ to empty array ④
	paus : 1000 ♥ ms
	set cross • invisible • 0
	call show
	0

if	num = • list • get value at length of array guesses • the	
	guesses - add value num to end	
Call	aropsprite num	
play	tone list • get value at num for 1 • beat until done •	
if	length of array guesses • 2 • length of array list • then	
pla	ay sound power up 🔹 until done 💌	
chi	ange score by 1	
cal	ll nextLevel	
$\odot$		
else	e	
set	showing - to true -	
chan	ge life by -1	
set	cross V invisible V OFF	
1	life v v then	
pla	sound power up • until done •	
set	guesses ▼ to empty array ⊕	
pau	use 1000 🔻 ms	
set	cross 🔹 invisible 🔹 🤇 🔍	
cal	L1 show	
Ð		

#### Step 11 – Keypresses

Now we need to hook up the key presses so the player can play back the tune. We will start with the up button.



Now add the down, left, right, A and B buttons so you have 6 in total like the below:

on up ▼ button released ▼	on down 🔻 button released 💌
if not showing  then	if not showing ▼ then
call guess 0	call guess 1
•	
on left ▼ button released ▼	on right 🔹 button released 💌
if not showing • then	if not showing  then
call guess 2	call guess 3
<b>O</b>	•
on B ▼ button released ▼	on A ▼ button released ▼
if not showing  then	if not showing • then
call guess 4	call guess 5

#### Extending the game

There are many ways that this game can be extended. Just a few ideas are given below.

- Animate the sprites as they fall.
- Rather than randomly generating a sequence for each level, add to the existing sequence to make it longer.
- Rather than randomly generating a sequence, use the notes to play a tune.