TASK 4: Finding neighboring weight vectors: Find the weight vectors that are neighbors to the winning weight vector. Common techniques for finding the neighboring vectors, include others hexagons, concentric squares, and the gaussian function.

TASK 4 Demo Outline: "Finding neighboring weight vectors"

```
(setf map '())   ;create an empty list
(create map 16) ;give the list 16-squared empty list elements
nil
(visualize map) ;displays the map-list as a grid
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
(setf map (init map random)) ;assigns each list in the map-list to have three elements a red,
green, and blue value RANDOMLY
(visualize map) ;displays the map-list as a grid
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
```

```
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(setf map (init map four-corners)) ;assigns each list in the map-list to have three elements a red,
green, and blue value to have red, green, blue, and black initialized at the four corners of the map
and have them fade away radially.
(visualize map) ;displays the map-list as a grid
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
```

(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(setf map (init map equidistant-from-center)) ;assigns each list in the map-list to have three elements a red, green, and blue value to have red, green, and blue initialized equidistant from the center of the map and have them fade away radially.
(visualize map) ;displays the map-list as a grid
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7

```
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125) (5 3 8) (5 6 7) (3 0 5) (7
6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(setf sample (create-sample random)) ;create a random sample to help select a winning RGB
weight vector
(print sample)
(32 245 0)
(setf sample (create-sample custom (0 245 32))) ;create a custom sample to help select a winning
RGB weight vector
(print sample)
(0 245 32)
(setf winning-weight-vector (winner map sample))
(print winning-weight-vector) ;prints the most similar RGB weight vector in the map to the one
given
(42 37 125)
(setf samples (create-samples 5 random)) ;creates 5 random samples using the (create-sample)
function, a specified amount of samples to generate, and a specifier to generate the sample RGB
vectors at random
(display samples) ;prints the most similar RGB weight vectors in the map to the ones given as a
list
((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0))
(setf samples (create-samples 5 custom ((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0
0)))) ;creates 5 custom samples by a prompt for manual input, a specified amount of samples to
generate, a specifier to generate the sample RGB vectors customly, and a list of lists containing
the 3 elements consisting of R G B values
(display samples) ;prints the most similar RGB weight vectors in the map to the ones given as a
list
```

((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0))
(setf sample (create-sample random)) ;create a random sample to help select a winning RGB weight vector
(print sample)
(32 245 0)
(setf sample (create-sample custom (0 245 32))) ;create a custom sample to help select a winning RGB weight vector
(print sample)
(0 245 32)
(setf winning-weight-vector (winner map sample euclidean-distance))
(print winning-weight-vector) ;prints the most similar RGB weight vector in the map to the one given
(42 37 125)
(setf samples (create-samples 5 random)) ;creates 5 random samples using the (create-sample) function, a specified amount of samples to generate, and a specifier to generate the sample RGB vectors at random
(setf winning-weight-vectors (winners map samples euclidean-distance)) ;sets the most similar RGB weight vectors in the map to the ones given as a list
(print winning-weight-vectors)
((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0))
(setf samples (create-samples 5 custom ((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0)))) ;creates 5 custom samples by a prompt for manual input, a specified amount of samples to generate, a specifier to generate the sample RGB vectors customly, and a list of lists containing the 3 elements consisting of R G B values
(setf winning-weight-vectors (winners map samples euclidean-distance)) ;sets the most similar RGB weight vectors in the map to the ones given as a list
(print winning-weight-vectors)
((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0))
(setf winning-weight-vector (winner map sample pearson-correlation-coefficient))
(print winning-weight-vector) ;prints the most similar RGB weight vector in the map to the one given
(42 37 125)
(setf samples (create-samples 5 random)) ;creates 5 random samples using the (create-sample) function, a specified amount of samples to generate, and a specifier to generate the sample RGB vectors at random
(setf winning-weight-vectors (winners map samples pearson-correlation-coefficient)) ;sets the most similar RGB weight vectors in the map to the ones given as a list
(print winning-weight-vectors)
((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0))

(setf samples (create-samples 5 custom ((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0)))) ;creates 5 custom samples by a prompt for manual input, a specified amount of samples to generate, a specifier to generate the sample RGB vectors customly, and a list of lists containing the 3 elements consisting of R G B values
(setf winning-weight-vectors (winners map samples pearson-correlation-coefficient)) ;sets the most similar RGB weight vectors in the map to the ones given as a list
(print winning-weight-vectors)
((42 37 125) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0))
(setf neighbors (find-neighbors map winning-weight-vector hexagons)) ;creates a list of lists consisting of the RGB vectors
(print neighbors)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(setf neighbors (find-neighbors map winning-weight-vector concentric-squares)) ;creates a list of lists consisting of the RGB vectors
(print neighbors)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)
(setf neighbors (find-neighbors map winning-weight-vector gaussian)) ;creates a list of lists consisting of the RGB vectors
(print neighbors)
(5 3 8) (5 6 7) (3 0 5) (7 6 0) (24 98 12) (34 32 87) (154 0 0) (42 37 125)