```
%%
% Ashley Crickard, arcricka
% arcricka@ncsu.edu
% 03-07-19
% Lab Section #202
% Project 2: Chaos Game, Spring 2019

%%
function [nextPoint] = restrictedOneAway(prevPoint, vertices, cutFraction)
% the restricted choas algorithm, where a vertex that is clockwise one
% vertex away can not be chosen. (clockwise or counterclockwise)

%Input: prevPoint - 2 by 1 vector of real numbers that shows the (x,y)
%                   coordinates of the previously chosen point.
%       vertices - number of vertices by 2 array, where each row is the coordinates
%                  of a specific vertex.
%                  For example, a square will be 4 by 2
%       cutFraction - a real number that is the calculated cutting fraction
%                  for a specific polygon.
%                  For example, a square would be 0.5.
%Return: nextPoint - 2 by 1 vector of the next point

% Creating the persistant variable assigned to randomVertex
persistent a

% getting a random number
randomNumber = randi([1,length(vertices)]);

if isempty(a)~=1

    % have to find the point and the point for the last number of the index
    % will have to be accounted for specially. This is for Clockwise.
    if randomNumber == 1
        c = vertices(length(vertices), :);
    else
        c = vertices(randomNumber - 1, :);
    end

    % have to find the point and the point for the last number of the index
    % will have to be accounted for specially. This is for counter Clockwise.
    if randomNumber == length(vertices)
        d = vertices(1, :);
    else
        d = vertices(randomNumber + 1, :);
    end

    % since a is not empty we must look to see if it is a pervious vertex
    % Error check for the clockwise and counter clockwise vertex
    while c == a | d == a
        % getting a random number
        randomNumber = randi([1,length(vertices)]);

        % have to find the point and the point for the last number of the index
        % will have to be accounted for specially. This is for Clockwise.
        if randomNumber == 1
            c = vertices(length(vertices), :);
        else
            c = vertices(randomNumber - 1, :);
```

```
        end

        % have to find the point and the point for the last number of the index
        % will have to be accounted for specially. This is for counter Clockwise.
        if randomNumber == length(vertices)
            d = vertices(1, :);
        else
            d = vertices(randomNumber + 1, :);
        end

    end
end

% Making a vertex out of the random number
randomVertex = vertices(randomNumber, :);

% Reassinging the value for the persistant varible z
a = randomVertex;

% Finding the next point
nextPoint = (prevPoint + randomVertex).*cutFraction;

end
```