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

%%
function [nextPoint] = restrictedPrev(prevPoint, vertices, cutFraction)
% the restricted choas algorithm, where a vertex that was previously chosen
% can not be selected agian for the next point

%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 z

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

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

% Using an if statement to make sure that we can error check for teh same
% point

if isempty(z)~=1
% since z is not empty, we are going to have to check it with the while
% loop

% Making sure the point was not used before; runs if the random vertex is
% equal to the previous vertex
while randomVertex == z
    % getting a random number
randomNumber = randi([1,length(vertices)]);

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

end % end of while
end % end of if


% Updating the value of the perisitant variable
z = randomVertex;


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

```
end
```