I noticed that the imported dataset is a matrix of multiple points. So, I will use MATLAB, which is good at matrix analysis, to solve this problem.

1. Import data

For code details, please kindly find the attached file " Data.m " (Open with MATLAB). For the convenience of review, I have copied all the code for importing data in the document as follows.

```
polygon = importdata('input_question_6_polygon',' ',0);
% ' ' is mean identify column separator as a space
points = importdata('input_question_6_points',' ',0);
% 0 is mean read from the beginning of line 0
vert_sum = size(polygon,1);
% Total number of vertices in the polygon.
vert_x = polygon(1:end,1);
% Matrix containing the x coordinates of the polygon's vertices.
vert_y = polygon(1:end,2);
% Matrix containing the y coordinates of the polygon's vertices.
point_x = points(1:end,1);
% Matrix containing the x coordinates of the test points.
point_y = points(1:end,2);
% Matrix containing the y coordinates of the test points.
```

2. PNPOLY

I will show the code of the PNPOLY firstly. Its method will describe in detail in the next section.

For code details, please kindly find the attached file " PNPOLY.m " (Open with MATLAB). For the convenience of review, I have copied all the code of the document as follows.

```
function [result] = PNPOLY(vert_sum,vert_x,vert_y,testpoint_x,testpoint_y)

result = 0;

for i = 1:vert_sum

    j = i-1;

    if j == 0
        j = vert_sum;
        % These steps are to control j is always equal to i - 1
        % But if i = 0 (j = 0), let j = nvert - 1
    end
```

```
    % The above steps simulate the connection of each point of the polygon
    % that is, simulate the formation of the edge
    % so as to prepare for the PNPOLY

    if ( ( testpoint_y > vert_y(i) ) ~= ( testpoint_y > vert_y(j) ) ) ...
           && ( ( testpoint_x - vert_x(i) ) < ( vert_x(j) - vert_x(i) ) ...
           * ( testpoint_y - vert_y(i) ) / ( vert_y(j) - vert_y(i) ) )
       result = ~result; % The core code of the PNPOLY
    end

end

% According to the PNPOLY so as to estimate the test point results
if result == 1
    result = 'inside';
else
    result = 'outside';
end

end
```

3. PNPOLY Method

A ray is drawn from the test point to the right. If the test point is inside the polygon, the ray has an odd number of intersections with the polygon. Otherwise, if the test point is outside the polygon, the ray has an even number of intersections with the polygon.

That is because, if the test point is inside, the first intersection will "rush out" the polygon, the second intersection will "enter" the polygon, and so on. Finally, the ray will "rush out" the polygon to leave an odd number of intersections. The same applies to the test point outside the polygon.

```
( testpoint_y > vert_y(i) ) ~= ( testpoint_y > vert_y(j) )
```

This sentence is meant to find the edge that might intersect the ray. We only need to limit the y-coordinate of one point of this edge is greater than that of the test point. The y-coordinate of the other point of this edge is less than that of the test point.

```
( testpoint_x - vert_x(i) ) < ( vert_x(j) - vert_x(i) ) ...
* ( testpoint_y - vert_y(i) ) / ( vert_y(j) - vert_y(i) )
```

This sentence is meant to determine whether the test point is on the left side of the edge.
Specifically, this statement is evolved from the following equation, which is to determine the point is on the left side of the edge

$$y - vert\_y[i] < \frac{vert\_y[j] - vert\_y[i]}{vert\_x[j] - vert\_x[i]} \cdot (x - vert\_x[i])$$

4. Export data

For tested data, please kindly find the attached file " output_question_6.txt ".
For code details, please kindly find the attached file " Data.m " (Open with MATLAB). For the convenience of review, I have copied all the code for exporting data in the document as follows.

```
fileID = fopen('output_question_6.txt','wt'); % Control the output

for i = 1:vert_sum
fprintf(fileID,'%u %u %s\n',point_x(i),point_y(i),PNPOLY(vert_sum,vert_x,vert_y,p
oint_x(i),point_y(i)));
% The test points are substituted into the algorithm one by one.
% Then, the function will write the results into the document
end

fclose(fileID); % Finish the control
```