HW12 M7

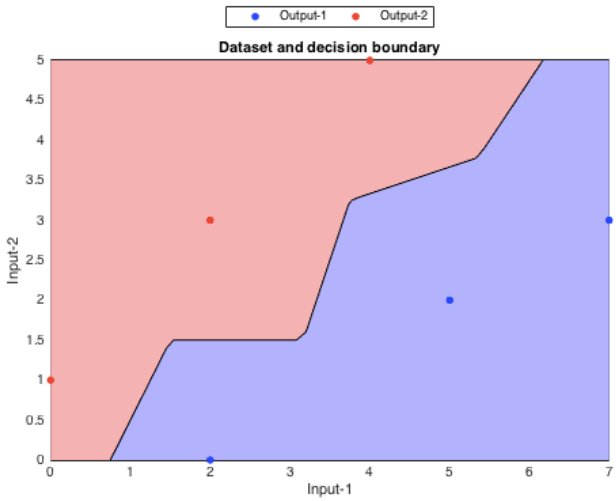
Answer (for reference only)

1.

Code:

```
DS.input = [2 0; 5 2; 7 3; 0 1; 2 3; 4 5]';
DS.output = [1 1 1 2 2 2];
DS.outputName = cell(1,2);
DS.outputName{1} = 'blue';
DS.outputName{2} = 'red';
%% Q1
figure;
knncPlot(DS, [], 'decBoundary');
```

Result:

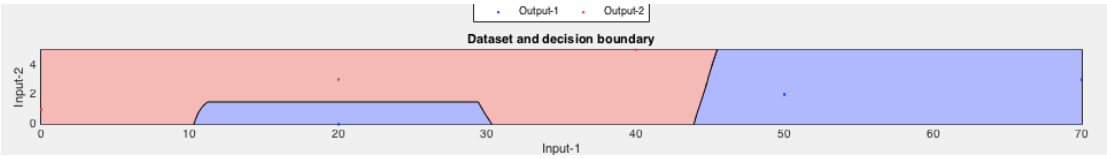


2.

Code:

```
%% Q2
DS2 = DS;
DS2.input(1,:) = DS2.input(1,:) * 10;
figure;
knncPlot(DS2, [], 'decBoundary');
```

Result:



Observation:

When the x-axis is multiple by 10, the values spread out more. We could see that the decision boundary is changed, and the influence of the x value is magnified. In order to avoid this problem, we should normalize (scale) the data before training or testing.

3.

Code:

```
%% Q3
trainSet = knncTrain(DS);
trainSet.k = 3;
computed = knncEval([1; 2], trainSet);
[n ~] = max(hist(computed));
fprintf('The output of 3-NNC of the point (1,2) is %s\n', DS.outputName{computed(n)})

DS.input = [DS.input [1 1;2 2]'];
DS.output = [DS.output 1 1];
trainSet = knncTrain(DS);
trainSet.k = 3;
computed = knncEval([1; 2], trainSet);
[n ~] = max(hist(computed));
fprintf('The output of adding points to training set is %s\n',
DS.outputName{computed(n)})
```

The result:

```
The output of 3-NNC of the point (1,2) is red
The output of adding points to training set is blue
```

Observation:

We could find that the output of testing the point (1,2) is 1, and it shows that the point is closer to the red region. And after we add the point (1,1) and point (2,2) to the blue region, 3-NNC would re-classify the point to the blue region.

4.

The complexity is O(nk+dk) when n is the number of data set points.

Distance computation requires O(nd) runtime. Perform O(n) work by looping through the training set observations, so the step overall requires O(nk) work. The sum of it is O(nd+nk).