Task 1: Image Preprocessing and Haar-like Feature Extraction
1. Resize all images in the "DogCat.zip" file to 48x48 pixels using the `imresize` function.
2. Compute the integral image for each resized image.
3. Calculate Haar-like features using the integral image with 30 boxes for each image.
4. Extract Haar-like features for both training and testing images, resulting in a 30-dimensional feature vector for each image.

Task 2: Classification using K-nearest neighbor (KNN) and Neural Networks (NN)
1. Train KNN and NN classifiers on the Haar-like features of the original 25 training images for each class (total of 50 images).
2. Test the trained KNN and NN models on the Haar-like features of the 15 testing images for each class (total of 30 images).
3. Evaluate the accuracy rates for each class using different K values (1, 3, 5, and 7).

Task 3: Manual Data Collection and Classification
1. Manually collect 30 additional training images for both classes (Dog and Cat).
2. Resize and crop the newly collected images to 48x48 pixels.
3. Train KNN and NN classifiers on the combined training dataset (original 25 + newly collected 30 images) for each class (total of 55 images).
4. Test the trained KNN and NN models on the testing

images and evaluate accuracy rates for each class using different K values (1, 3, 5, and 7).

Code for Problem 1 and 2

```
clear all;
close all;
clc;
 img = imread('DogCat/Testing/Cat/10.jpg');
figure,imshow(img);
img_gray = rgb2gray(img);
figure,imshow(img_gray);
img_gray = imresize(img_gray,[48 48]);
figure,imshow(img_gray);
haar = zeros(2,1);

haar=zeros(2,1);

rec1=[1,1,32,64];
rec2=[33,1,64,64];

sum_w=0;
for i=rec1(1):rec1(3)
    for j=rec1(2):rec(4)
        sum_w=sum_w+double(img_gray(i,j));
    end
end


sum_b=0
for i=rec2(1):rec2(3)
    for j=rec2(2):rec2(4)
```

```
        sum_=sum_b+double(img_gray(i,j));
    end
end
haar(1)=(sum_w-sum_b)/(48*48);

rec1=[1,1,64,32];
rec2=[1,33,64,64];

sum_w=0;
for i=rec1(1):rec1(3)
    for j=rec1(2):rec(4)
        sum_w=sum_w+double(img_gray(i,j));
    end
end



sum_b=0
for i=rec2(1):rec2(3)
    for j=rec2(2):rec2(4)
        sum_=sum_b+double(img_gray(i,j));
    end
end
haar(2)=(sum_w-sum_b)/(48*48);
disp(haar);
```

Code for Problem 4

```
disp('Preparing training data');
folderCat = './DogCat/Testing/Cat/';
folderDog = './DogCat/Testing/Dog/';
filesCat = dir(fullfile(folderCat, '*.jpg'));
filesDog = dir(fullfile(folderDog, '*.jpg'));
```

```
feats = zeros(length(filesCat) + length(filesDog), 256);
labels = zeros(length(filesCat) + length(filesDog),1);

for i = 1:length(filesCat)
disp(i);
filename = filesCat(i,1).name;
img = imread([folderCat filename]);
img = imresize(img,[256,256]);
feat = lbp(img);
feats(i,:) = feat;
labels(i) = 0;
end

for i = 1:length(filesDog)

    disp(i);
    filename = filesDog(i,1).name;
    img = imread([folderDog filename]);
    img = imresize(img,[256,256]);
    feat = lbp(img);
    feats(i + length(filesCat),:) = feat;
    labels(i + length(filesCat)) = 1;
end

disp('Preparing testing data');
folderCat = './DC/Testing/Cat/';
folderDog = './DC/Testing/Dog/';
filesCat = dir(fullfile(folderCat, '*.jpg'));
filesDog = dir(fullfile(folderDog, '*.jpg'));
featsTest = zeros(length(filesCat) + length(filesDog),
256);
groundtruthLabel = zeros(length(filesCat) +
```

```
length(filesDog), 1);
predictedLabel = zeros(length(filesCat) +
length(filesDog), 1);

for i = 1:length(filesCat)
disp(i);
filename = filesCat(i,1).name;
img = imread([folderCat filename]);
img = imresize(img,[256,256]);
feat = lbp(img);
featsTest(i,:) = feat;
groundtruthLabel(i) = 0;
end

for i = 1:length(filesDog)
    disp(i);
    filename = filesDog(i,1).name;
    img = imread([folderDog filename]);
    img = imresize(img,[256,256]);
    feat = lbp(img);
    featsTest(i + length(filesCat),:) = feat;
    groundtruthLabel(i + length(filesCat)) = 1;
end
disp('Performing training');
% [w,b,pass] = PerceptronTrain(feats,labels');

disp('Performing testing');
accurateClassification = 0;
for i = 1:size(featsTest,1)
feat = featsTest(i,:);
ey = feat*w+b; % apply the perceptron classification rule
if ey>=0.5
```

```
predictedLabel(i) = 1;
else
predictedLabel(i) = 0;
end
if(predictedLabel(i) == groundtruthLabel(i))
accurateClassification = accurateClassification + 1;
end
end
```