Assignment 2 , Shahriar Shayesteh

In [ ]:
```
from PIL import Image, ImageDraw
import numpy as np
import math
from scipy import signal
import ncc
import PIL
import scipy
```

Question 2

In [2]:
```
#each time apply the given scale on weight and height of the image and returr
def height_width_tuple(weight,height):
    a=  int(weight*0.75)
    b = int(height*0.75)
    return (a,b)

#from a given image, it makes a pyramid of it. each time scale the image unti
def MakePyramid(image, minsize):

    #make a list named Pyramid
    pyramid = []
    #open the given image and append it to the list and then convert it to a
    image = Image.open(image)
    img = np.asarray(image).astype('float')
    pyramid.append(image)

    #get the image weight and height
    weight = img.shape[0]
    height = img.shape[1]
    img1 = img

    while(minsize < min(img1.shape)):

        #find a scaled weight,height of the given image
        size=height_width_tuple(weight,height)
        weight = size[0]
        height = size[1]


        #make the scaled image and append it to the pyramid list
        img11= image.resize((weight,height),PIL.Image.BICUBIC)
        img1 = np.asarray(img11).astype('float')
        scaled_image = Image.fromarray(img1)
        pyramid.append(img11)


    return pyramid
```

In [4]:
```
pyramid1 = MakePyramid("family.jpg", 10)
len(pyramid1)
```

Out[4]: 13

In [5]:
```
pyramid2 = MakePyramid("family.jpg", 109)
len(pyramid2)
```

Out[5]: 5

In [6]:
```
pyramid2 = MakePyramid("fans.jpg", 88)
len(pyramid2)
```

Out[6]: 5

Question 3

In [7]:
```
# it shows pyramid of images of given pyramid list.
def ShowPyramid(pyramid):

    # using geometric peogression, we can calculate the lenght of
    #the background imagebased on the lenght of the pyramid list
    backgroung_weight = int(np.ceil( (1 - (0.75)**(len(pyramid)))/(0.25)))
    bacckground = Image.new("L", ((backgroung_weight)*pyramid[0].size[0],pyra


    x1,y1= (pyramid[0].size[0],pyramid[0].size[1])
    bacckground.paste(pyramid[0].convert('RGB') ,(0,0,x1,y1))

    #we loop through all the images in pyramid list from the last element to
    for i in range(1,len(pyramid)):
    # last image in pyramid pasted as the first image
    #in the background and then it goes through all images
    # for the first image we start from (0,0) to the lenght of image,
    #for the nex ones we consider place of all last images pasted on the back
        if(i != 1 ):
            x1 = x1+ pyramid[i-1].size[0]
            y1 = y1+ pyramid[i-1].size[1]


        x,y= pyramid[i].size

        bacckground.paste(pyramid[i].convert('RGB') ,(x1,0,x1+x,y))


    bacckground.save('pyramid.jpg')
    return Image.open('pyramid.jpg').convert('L')
```

In [12]:
```
pyramid1= ShowPyramid(MakePyramid("judybats.jpg", 10))
pyramid2= ShowPyramid(MakePyramid("family.jpg", 100))
pyramid3= ShowPyramid(MakePyramid("students.jpg", 200))
pyramid4= ShowPyramid(MakePyramid("fans.jpg", 88))
```

In [13]:
```
pyramid1
```

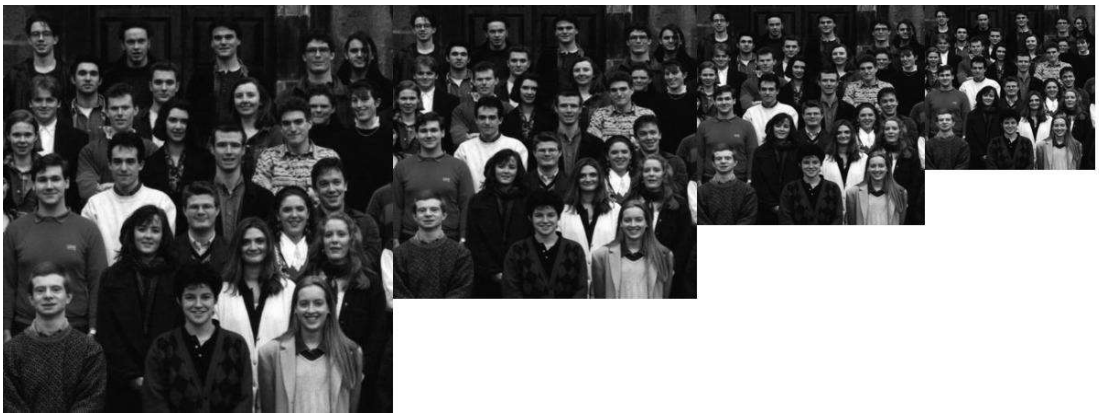
Out[13]:

In [14]:
```
pyramid2
```

Out[14]:

In [15]:
```
pyramid3
```

Out[15]:



In [16]:
```
pyramid4
```

Out[16]:

Question 4

In [18]:

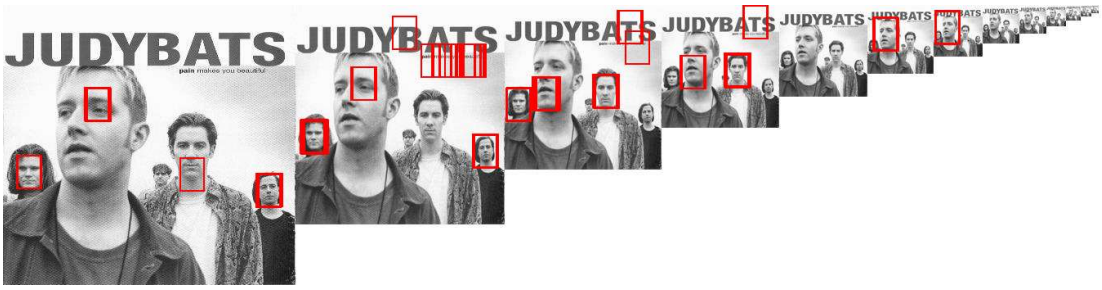
```
#a function, if it gets a pyramid,
#it will apply template on each image
def FindTemplate(image, template, threshold):

    template = Image.open(template)

    #apply a gussian filter on template based on the imgae size to get a bett
    template1 = scipy.ndimage.filters.gaussian_filter(template, sigma =3)

    #change the template size to given size in the question for original imag
    #and sclaed version of it based on i
    temp_size = (15,int((15*50)/37))
    template1 = template.resize(temp_size, Image.BICUBIC)

    # apply normalized autocorrelation of template on the image to find match
    template_finder = ncc.normxcorr2D(image, template1)
    template_finder = np.transpose(template_finder)
    template_finder = template_finder.reshape((image.size[0], image.size[1]))
    image = image.convert('RGB')

    #If the array's elements are less than
    #threshold they're going to be 0 and else 1
    for x in range(0,template_finder.shape[0]):
        for y in range(0,template_finder.shape[1]):

            if(template_finder[x][y] >= threshold):
                template_finder[x][y] = 1

            if(template_finder[x][y] < threshold):
                template_finder[x][y] = 0

    #find the location of elements which are not 0, and draw a rectangle arou
    for x in range(0,template_finder.shape[0]):
        for y in range(0,template_finder.shape[1]):

            if(template_finder[x][y] == 1):
                x1 = x-int(temp_size[0])
                x2 = x+int(temp_size[0])


                y1 = y-int(temp_size[1])
                y2 = y+int(temp_size[1])

                draw = ImageDraw.Draw(image)
                draw.line((x1,y1,x2,y1),fill="red",width=2)
                del draw

                draw = ImageDraw.Draw(image)
                draw.line((x1,y1,x1,y2),fill="red",width=2)
                del draw

                draw = ImageDraw.Draw(image)
                draw.line((x1,y2,x2,y2),fill="red",width=2)
                del draw
```

```
                draw = ImageDraw.Draw(image)
                draw.line((x2,y2,x2,y1),fill="red",width=2)
                del draw

    image.save('111.jpg' )

    return image.convert('RGB')
```

In [21]:
```
FindTemplate(pyramid1,"template.jpg",0.5)
```

```
C:\Users\Shahriar\ComputerVision\assignment2\ncc.py:59: RuntimeWarning: div
ide by zero encountered in true_divide
  nxcorr = np.where(denom < tol, 0, numer/denom)
```

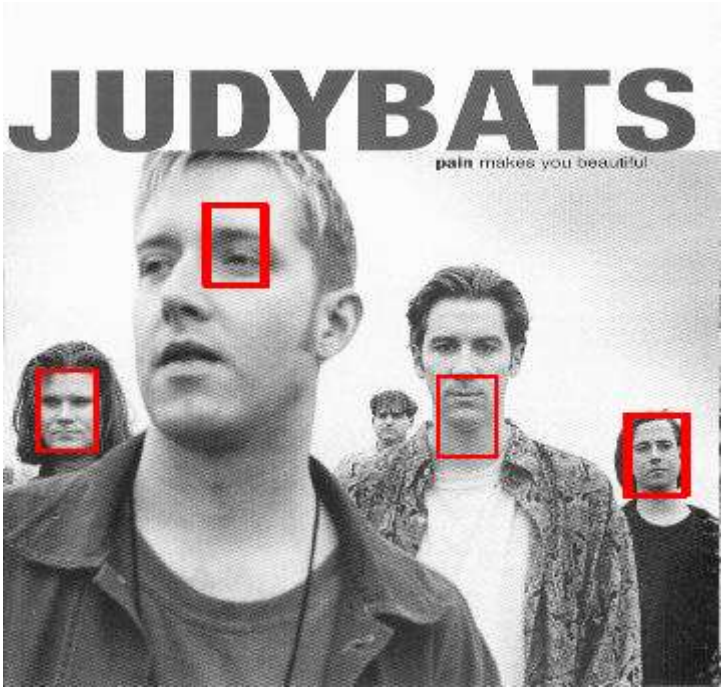
Out[21]:



In [1077]:
```
image = Image.open('judybats.jpg')

FindTemplate(image,"template.jpg",0.5)
```

```
C:\Users\Shahriar\ComputerVision\assignment2\ncc.py:59: RuntimeWarning: div
ide by zero encountered in true_divide
  nxcorr = np.where(denom < tol, 0, numer/denom)
```

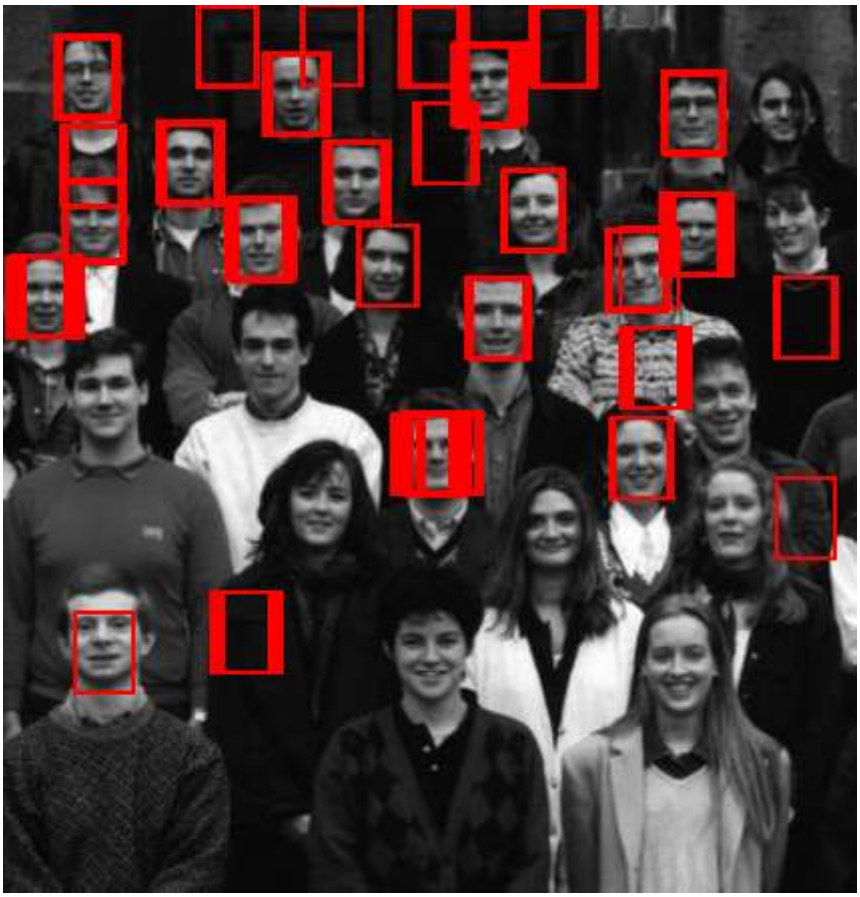
Out[1077]:



Question5

In [1079]:
```
image = Image.open('students.jpg')

FindTemplate(image, "template.jpg",0.5)
```

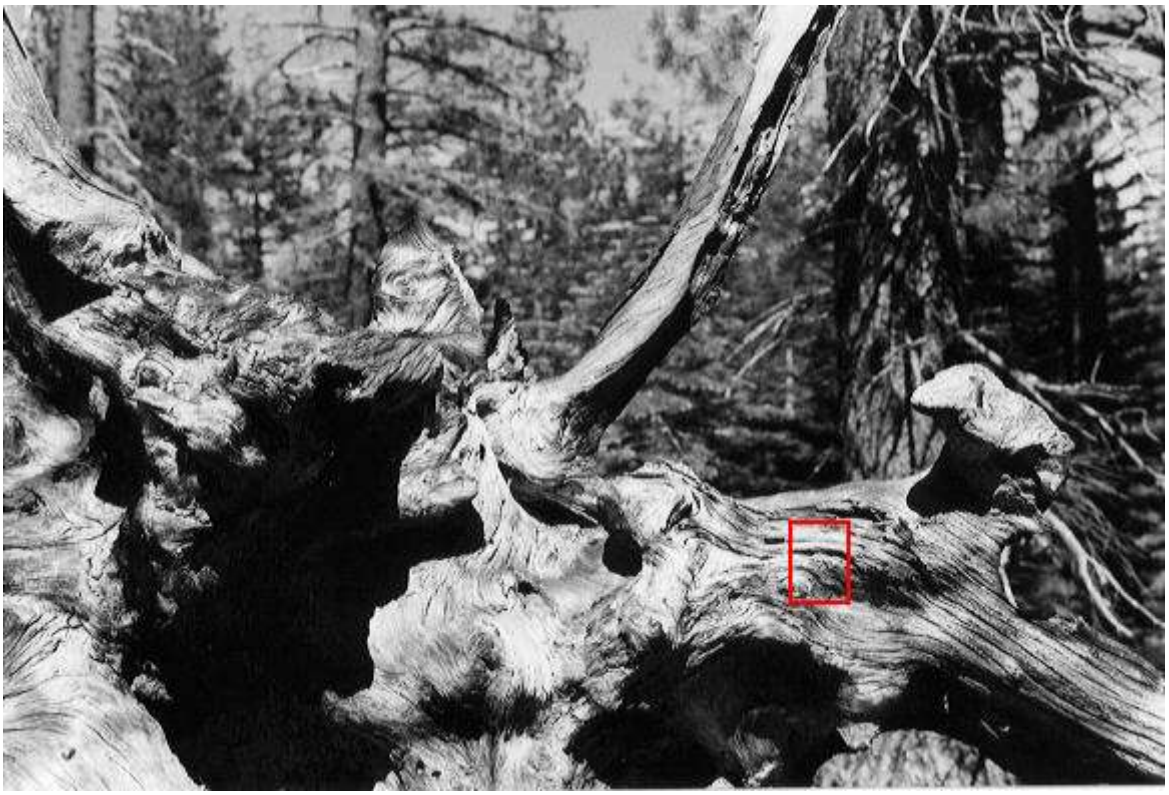
Out[1079]:



```
False positive = 9 , False negative = 10
```

In [1091]:
```
image = Image.open('tree.jpg')
FindTemplate(image , "template.jpg", 0.6)
```

Out[1091]:



```
False positive = 1 , False negative = 0
```

In [1081]:
```
image = Image.open('family.jpg')
FindTemplate(image, "template.jpg", 0.5)
```

```
C:\Users\Shahriar\ComputerVision\assignment2\ncc.py:59: RuntimeWarning: div
ide by zero encountered in true_divide
  nxcorr = np.where(denom < tol, 0, numer/denom)
```

Out[1081]:

```
False positive = 2 , False negative = 2
```

In [1082]:
```
image = Image.open('fans.jpg')

FindTemplate(image, "template.jpg", 0.53)
```

```
C:\Users\Shahriar\ComputerVision\assignment2\ncc.py:59: RuntimeWarning: div
ide by zero encountered in true_divide
  nxcorr = np.where(denom < tol, 0, numer/denom)
```

Out[1082]:

```
False positive = 2 , False negative = 2
```

In [1083]:
```
image = Image.open('sports.jpg')

FindTemplate(image, "template.jpg", 0.484)
```

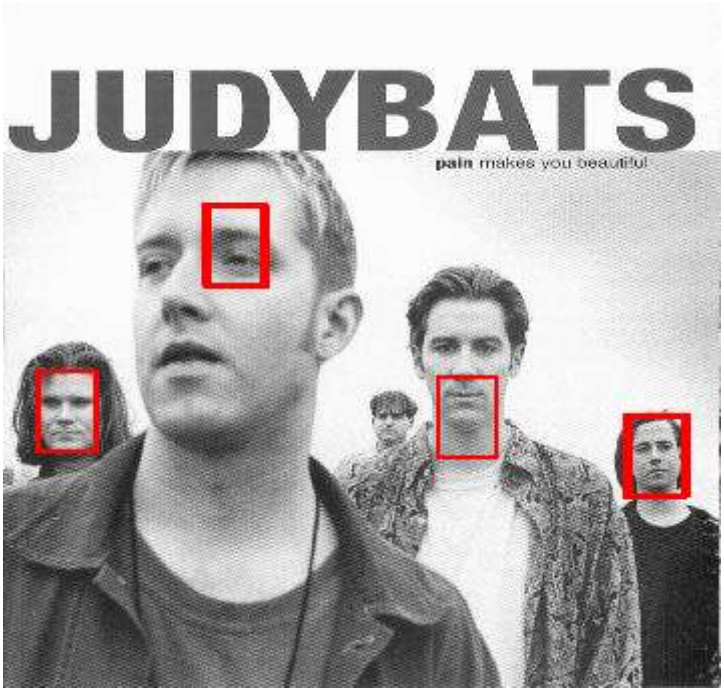
Out[1083]:


```
False positive = 2 , False negative = 2
```

In [22]:
```
image = Image.open("judybats.jpg")

FindTemplate(image, "template.jpg", 0.5)
```

```
C:\Users\Shahriar\ComputerVision\assignment2\ncc.py:59: RuntimeWarning: div
ide by zero encountered in true_divide
  nxcorr = np.where(denom < tol, 0, numer/denom)
```

Out[22]:




```
False positive = 2 , False negative = 2
```

Question 5

Recall = (True-Positive)/(False-Negative + True-Positive)

Precision = (True-Positive)/(False-Positive + True-Positive)

1) JudyBats => Recall = 3/(2+3) = 0.6 /\ Precision = 3/(1+3)= 0.75

2) sports => Recall = 0/(2+0) = 0 /\ Precision = 0/(2+0)= 0

3) students => Recall = 17/(10+17) = 0.63 /\ Precision = 17/(9+17)= 0.65

4) tree => Recall = it's not defined /\ Precision = 0/(1+0)= 0

5) fans => Recall = 1/(1+2) = 0.33 /\ Precision = 1/(1+2)= 0.33

6) family => Recall = 1/(1+2) = 0.33 /\ Precision = 1/(1+2)= 0.33

This method works only when there are some areas on the image having very similar size and shape and direction and intensity with the given template. In our problem, template is face of a man that was cut from the judybats image and our goal is to find and match this template in different given images,although result would not show a reliable method for doing a robust face detection for instance: in some pictures there is no face in it like tree.jpg, or faces are in different sizes and direction like sport.jpg and fans.jpg. As a result, the outputs will not be desireable and we cannot have a good performance with this method, and the reason that we have a very low recall rate on some images,I think, is that always it should be some parts in a picture ;which gets correlated to the template; close to template in term of scale, shape, orientation, and intensity, and a difference in even one of these elements can make the prediction difficult and unrobust.

In [ ]: