The Project

1. This is a project with minimal scaffolding. Expect to use the the discussion forums to gain insights! It's not cheating to ask others for opinions or perspectives!
2. Be inquisitive, try out new things.
3. Use the previous modules for insights into how to complete the functions! You'll have to combine Pillow, OpenCV, and Pytesseract
4. There are hints provided in Coursera, feel free to explore the hints if needed. Each hint provide progressively more details on how to solve the issue. This project is intended to be comprehensive and difficult if you do it without the hints.

The Assignment

Take a ZIP file (https://en.wikipedia.org/wiki/Zip_(file_format)) of images and process them, using a library built into python (https://docs.python.org/3/library/zipfile.html) that you need to learn how to use. A ZIP file takes several different files and compresses them, thus saving space, into one single file. The files in the ZIP file we provide are newspaper images (like you saw in week 3). Your task is to write python code which allows one to search through the images looking for the occurrences of keywords and faces. E.g. if you search for "pizza" it will return a contact sheet of all of the faces which were located on the newspaper page which mentions "pizza". This will test your ability to learn a new (library (https://docs.python.org/3/library/zipfile.html)), your ability to use OpenCV to detect faces, your ability to use tesseract to do optical character recognition, and your ability to use PIL to composite images together into contact sheets.

Each page of the newspapers is saved as a single PNG image in a file called images.zip (./readonly/images.zip). These newspapers are in english, and contain a variety of stories, advertisements and images. Note: This file is fairly large (~200 MB) and may take some time to work with, I would encourage you to use small_img.zip (./readonly/small_img.zip) for testing.

Here's an example of the output expected. Using the small_img.zip (./readonly/small_img.zip) file, if I search for the string "Christopher" I should see the following image:

```
Results found in file a-0.png
```

```
Results found in file a-3.png
```

If I were to use the images.zip (./readonly/images.zip) file and search for "Mark" I should see the following image (note that there are times when there are no faces on a page, but a word is found!):

Results found in file a-0.png

Results found in file a-1.png

Results found in file a-10.png
But there were no faces in that file!
Results found in file a-13.png

Results found in file a-2.png

Results found in file a-3.png

Results found in file a-8.png
But there were no faces in that file!

Note: That big file can take some time to process - for me it took nearly ten minutes! Use the small one for testing.

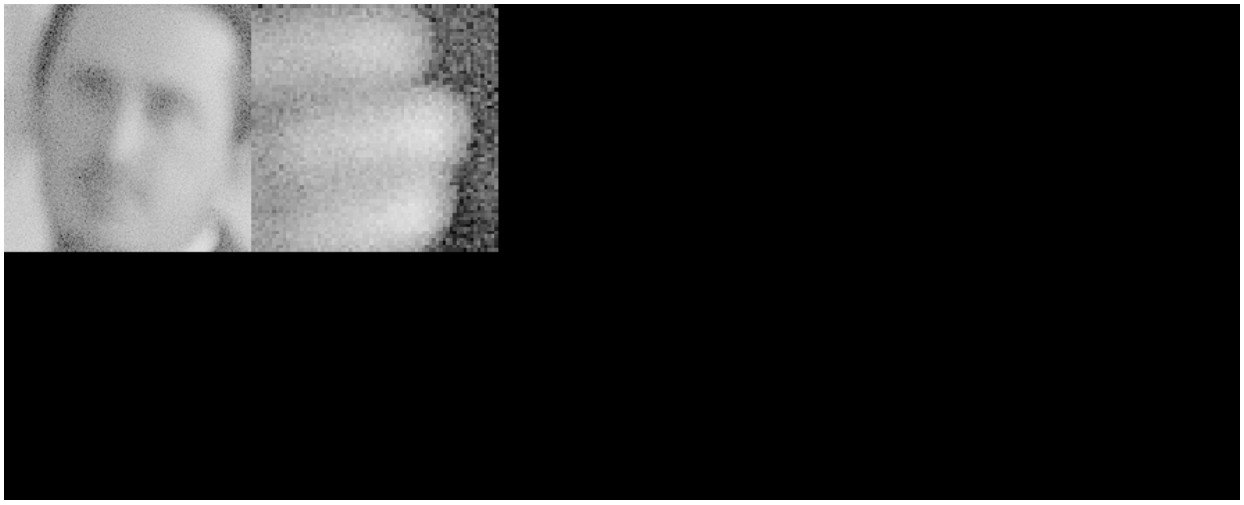
In [1]:
```
import zipfile

from PIL import Image
import pytesseract
import cv2 as cv
import numpy as np

# loading the face detection classifier
face_cascade = cv.CascadeClassifier('readonly/haarcascade_frontalface_
default.xml')

#loading images
def loadImages(file_ref):
    archive=zipfile.PyZipFile(file_ref)
    file_refs=archive.namelist()

    images=[]

    for ref in file_refs:
        file=archive.open(ref)
        img = Image.open(file)
        images.append(img)
    archive.close()
    return (images, file_refs)

#detect faces in image
def getFaces(image):
    # And we'll convert it to grayscale using the cvtColor image
    gray = cv.cvtColor(np.asarray(image), cv.COLOR_BGR2GRAY)
    rectangles = face_cascade.detectMultiScale(gray, 1.35)
    faces=[image.crop((rec[0],rec[1],rec[0]+rec[2],rec[1]+rec[3])) for
rec in rectangles]
    return faces

#create contact sheet with images
def createContactsheet(images):
    # create a contact sheet
    first_image=images[0]
    cols=5
    rows=2
    contact_sheet=Image.new(first_image.mode, (first_image.width*cols,
first_image.height*rows))
    x=0
    y=0

    for img in images:
    # Lets paste the current image into the contact sheet
        new_img=img.resize((first_image.width,first_image.height))
        contact_sheet.paste(new_img, (x, y) )
        # Now we update our X position. If it is going to be the width
```

```
of the image, then we set it to 0
        # and update Y as well to point to the next "line" of the cont
act sheet.
        if x+first_image.width == contact_sheet.width:
            x=0
            y=y+first_image.height
        else:
            x=x+first_image.width

    # resize and display the contact sheet
    #contact_sheet = contact_sheet.resize((int(contact_sheet.width/2),
int(contact_sheet.height/2) ))
    display(contact_sheet)

#create contactsheets for images where keyword is found
def createContactsheets(file, keyword):
    (images, file_refs)=loadImages(file)
    for i, image in enumerate(images):
        if keyword in pytesseract.image_to_string(image).split():
            print("Results found in file {}".format(file_refs[i]))
            faces=getFaces(image)
            if(len(faces)==0):
                print("But there were no faces in that file!")
            else:
                createContactsheet(faces)

data_src=["small_img.zip", "images.zip"]
keywords=["Christopher", "Mark"]
```

In [6]:
```
createContactsheets(data_src[0], keywords[0])
```

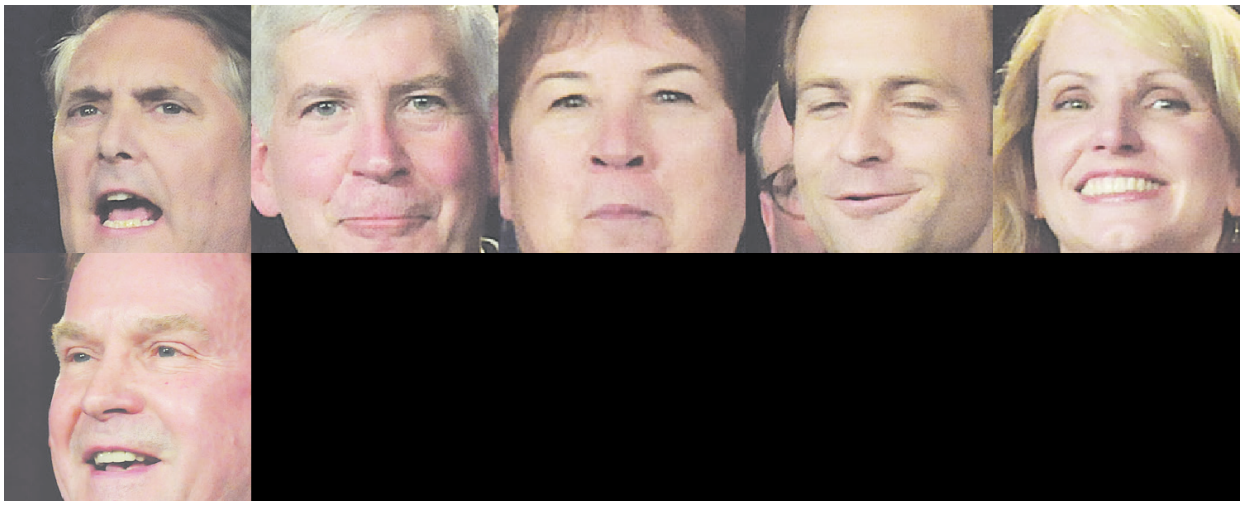
```
Results found in file a-0.png
```

```
Results found in file a-3.png
```



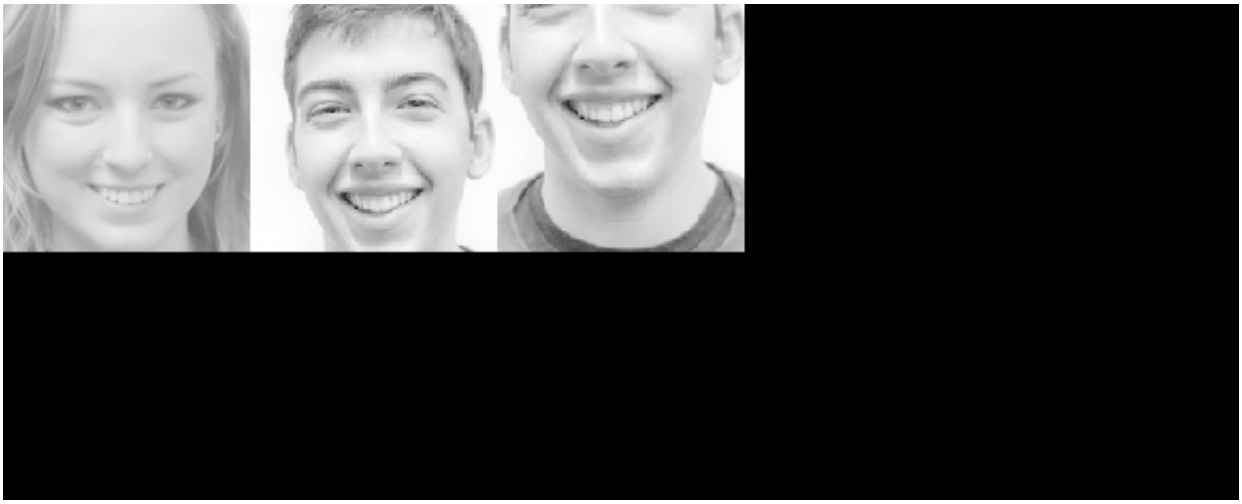
In [2]:
```
createContactsheets(data_src[1], keywords[1])
```

```
Results found in file a-0.png
```

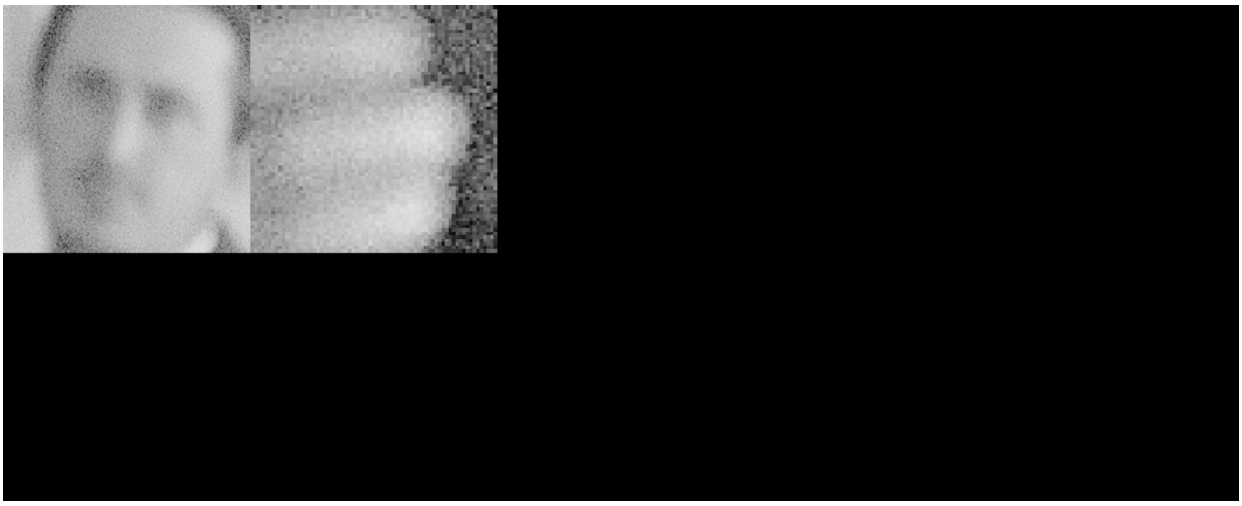


```
Results found in file a-1.png
```

Results found in file a-2.png



Results found in file a-3.png



Results found in file a-8.png

In [ ]: