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:

Christopher Search

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!):

Mark Search

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

In [2]:

```
import PIL
import zipfile
from IPython.display import display
from PIL import Image
#from PIL import ImageDraw
import pytesseract
import cv2 as cv
import numpy as np
import os

# loading the face detection classifier
face_cascade = cv.CascadeClassifier('readonly/haarcascade_frontalface_default.xm
l')

# the rest is up to you!
# Open the zip file
#unzip = zipfile.ZipFile("readonly/small_img.zip","r")
```

In [3]:

```
def unzip(file):
    print(file)
    unzip = zipfile.ZipFile("{}".format(file),"r")
    foldername = file.split('/')[1].split('.')[0]
    unzip.extractall('{}'.format(foldername))
    ## so all the contents in readonly/small_img.zip will be saved in the folder
small_img
    ## and all the contents in readonly/images.zip will be saved in the folder i
mages
    return foldername
```

Creating a Global List

In [12]:

```
def global_list(folder):
    page_list = os.listdir(folder)
    print(page_list)
    globallist = []
    for page_name in page_list:
        print(page_name)
        # Load text via pytesseract
        img = Image.open("{}/{}".format(folder,page_name))
        try:
            text = pytesseract.image_to_string(img)
            # Separate text into individual words, and split it into a list stor
e each word as an item in list for easy indexing.
            text = text.replace("-\n","").replace("\n", " ").split()
        except:
            pass
        # To detect faces
        cvimg = cv.imread("{}/{}".format(folder,page_name))
        try:
            gray = cv.cvtColor(cvimg, cv.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray,1.3,5).tolist()
        except:
            faces = []
            pass
        # To set bounding box parameters.
        bbox_params = []
        for box in faces:
            bbox_params.append([box[0],box[1],box[0]+box[2], box[1]+box[3]])
        # Upload all attributes of this newspaper image to the global list.
        # Each page in the globallist is hence represented by a dictionary
        globallist.append({'filename':'{}'.format(page_name), 'text':text, 'bbo
x':bbox_params})
    # For debugging
    test = [(item['filename'],item['bbox']) for item in globallist]
    print(test)
    # Return the globallist
    return globallist
```

Writing a function that will create a contact sheet and align images

Now that the bounding box for images are already loaded into the globallist dict, we can pass the globallist into the the `searchresults()` that will accept a search parameter.

In [13]:

```
def searchresults(param,zippfile):
    # First search for those words in text
    folder = unzip(zippfile)
    globallist = global_list(folder)
     # First search for those words in text
    print("---------------------------------------------")
    for image in globallist:
        if param in image['text']:
            print("Results found in file ", image['filename'])
            # Produce contact sheet for each result found here if there are pict
ures of faces.
            if (image['bbox'] == []):
                print("But there were no faces in that file!")
            else:
                # Now create the image
                face_list = []
                img = Image.open(image['filename'])
                for box in image['bbox']:
                    face_list.append(img.crop(box))

                # Create the contact sheet
                contact_sheet= Image.new(img.mode, (500,100*(int(np.ceil(len(fac
e_list)/5)))))
                # Define the starting coordinates
                x = 0; y = 0
                for face in face_list:
                    face.thumbnail((100,100))
                    contact_sheet.paste(face, (x,y))
                    if x+100 == contact_sheet.width: ## At this point this is at
the right border of the contactsheet
                        x = 0; y+=100
                    else:
                        x+=100
                #No need to resize the contact sheet in this case
                display(contact_sheet)

        else:
            pass
```

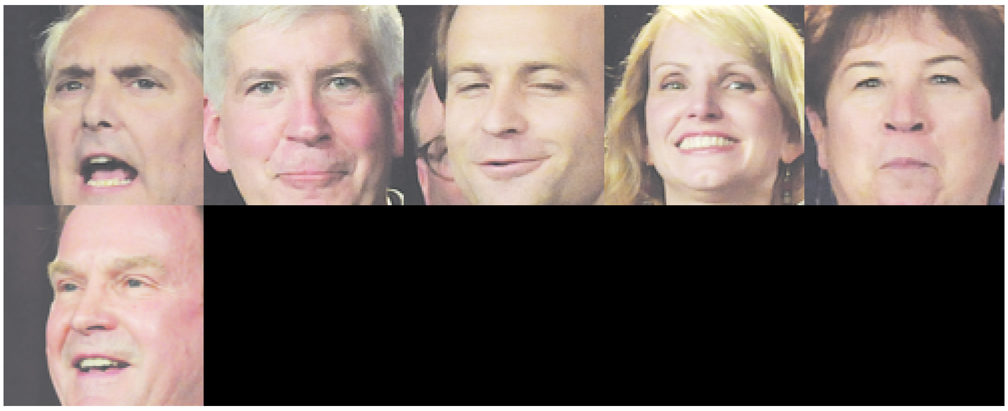
Test

In [10]:

```
searchresults("Christopher","readonly/small_img.zip")
```

```
readonly/small_img.zip
['a-3.png', 'a-1.png', 'a-0.png', 'a-2.png']
a-3.png
a-1.png
a-0.png
a-2.png
[('a-3.png', [[1805, 1403, 1985, 1583], [1936, 1781, 2005, 1850]]),
('a-1.png', [[492, 1366, 618, 1492], [833, 2382, 1045, 2594], [2235,
2448, 2286, 2499], [2063, 2499, 2114, 2550], [2515, 2419, 2576, 248
0]]), ('a-0.png', [[3138, 1733, 3419, 2014], [1967, 1882, 2183, 209
8], [1152, 2001, 1361, 2210], [1665, 2017, 1889, 2241], [2544, 1955,
2747, 2158], [2653, 3055, 2954, 3356]]), ('a-2.png', [[2104, 709, 22
18, 823], [661, 1542, 837, 1718]])]
---------------------------------------------
Results found in file  a-3.png
```

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

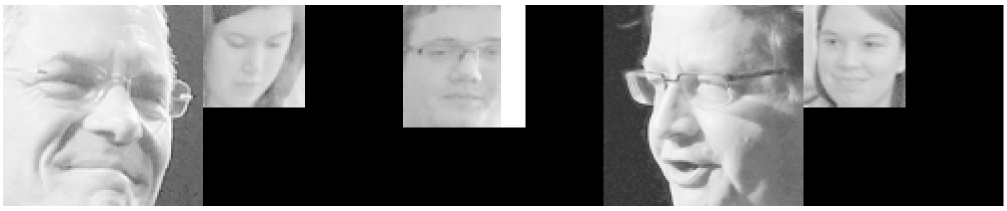


In [14]:

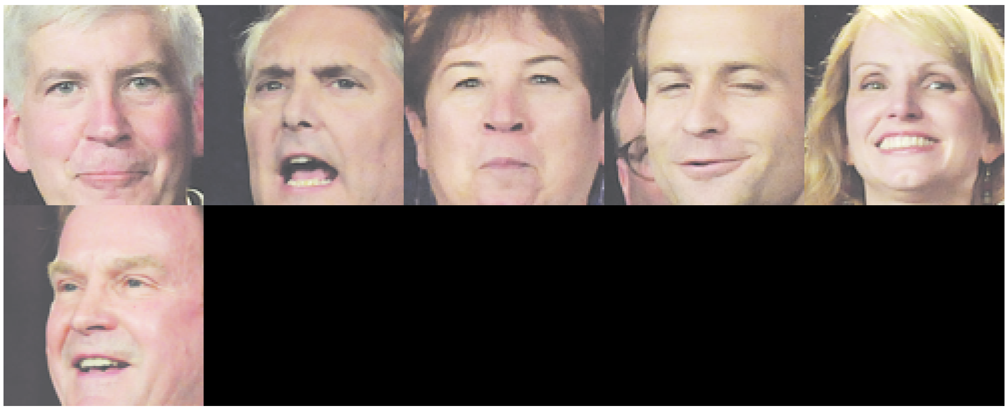
```
searchresults("Mark","readonly/images.zip")
```

```
readonly/images.zip
['a-7.png', 'a-12.png', 'a-3.png', 'a-10.png', 'a-1.png', 'a-9.png',
'a-5.png', 'a-0.png', 'a-8.png', 'a-13.png', 'a-4.png', 'a-11.png',
'a-2.png', 'a-6.png']
a-7.png
a-12.png
a-3.png
a-10.png
a-1.png
a-9.png
a-5.png
a-0.png
a-8.png
a-13.png
a-4.png
a-11.png
a-2.png
a-6.png
[('a-7.png', []), ('a-12.png', []), ('a-3.png', [[1805, 1403, 1985,
1583], [1936, 1781, 2005, 1850]]), ('a-10.png', []), ('a-1.png', [[4
92, 1366, 618, 1492], [2235, 2448, 2286, 2499], [2515, 2419, 2576, 2
480], [833, 2382, 1045, 2594], [2063, 2499, 2114, 2550]]), ('a-9.pn
g', [[761, 1298, 1268, 1805]]), ('a-5.png', [[2222, 834, 2294, 906],
[2106, 852, 2162, 908], [728, 720, 797, 789], [1177, 890, 1236, 94
9], [1433, 891, 1491, 949], [2384, 5215, 2482, 5313]]), ('a-0.png',
[[1967, 1882, 2183, 2098], [3138, 1733, 3419, 2014], [2544, 1955, 27
47, 2158], [1152, 2001, 1361, 2210], [1665, 2017, 1889, 2241], [265
3, 3055, 2954, 3356]]), ('a-8.png', []), ('a-13.png', [[2261, 2192,
2319, 2250]]), ('a-4.png', []), ('a-11.png', []), ('a-2.png', [[210
4, 709, 2218, 823], [661, 1542, 837, 1718]]), ('a-6.png', [[2506, 72
3, 3013, 1230], [2565, 2671, 2675, 2781], [242, 5764, 304, 5826]])]
---------------------------------------------
Results found in file  a-3.png
```

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



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



```
Results found in file  a-8.png
But there were no faces in that file!
Results found in file  a-2.png
```