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 [1]:
```python
import zipfile

from PIL import Image
from PIL import ImageDraw
import pytesseract
import cv2 as cv
import numpy as np
import pytesseract
from collections import defaultdict

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

# the rest is up to you!
```

In [ ]:
```python
dict_file = defaultdict(defaultdict)
```

```
In [2]: #Load one test image
        zips = zipfile.ZipFile('readonly/small_img.zip')
```

```
In [3]: file_names_in_zip = zips.namelist()
```

```
In [4]: file_names_in_zip
```

```
Out[4]: ['a-0.png', 'a-1.png', 'a-2.png', 'a-3.png']
```

Get image from file

Process:

```
- File to faces
- Faces to list
- list to dictionary
```

```
In [79]: def parse_image_from_file(file_name_in_zip, zips, dict_file):

             img_extracted = zips.extract(file_name_in_zip)

             img_pil = Image.open(img_extracted)

             img = cv.imread(img_extracted)
             # And we'll convert it to grayscale using the cvtColor image
             gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

             faces = face_cascade.detectMultiScale(gray, 1.3) #1.3

             for x,y,w,h in faces:

                 #drawing.rectangle((x,y,x+w,y+h), outline="red")
                 #display(pil_img)

                 dict_file[file_name_in_zip]['image'].append(img_pil.crop((x,y,x+w,y+h)))
             return dict_file
```

```
In [80]: for file_name_in_zip in file_names_in_zip:
             dict_file[file_name_in_zip]['image'] = []
             dict_file = parse_image_from_file(file_name_in_zip, zips, dict_file)
```

```
In [77]: #dict_file
```

Get text from file

Process:

- read image to string
- string split by word
- strip words

In [37]:
```
def parse_text_from_file(file_name_in_zip, zips, dict_file):
    img_extracted = zips.extract(file_name_in_zip)
    img_pil = Image.open(img_extracted)
    img_pil = img_pil.convert("L")
    # and run OCR
    text = pytesseract.image_to_string(img_pil)
    for i in text.split('\n'):
        if len(i)>3:
            for j in i.split(' '):
                dict_file[file_name_in_zip]['text'].append(j.strip().upper())

    return dict_file
```

In [38]:
```
for file_name_in_zip in file_names_in_zip:
    dict_file[file_name_in_zip]['text'] = []
    dict_file = parse_text_from_file(file_name_in_zip, zips, dict_file)
```

In [101]:
```
#dict_file
```

Search file

Process:

```
- Enter a String
- Convert to string upper()
- For dictionary with string
    If have string:
        put the image in a list
- Print
    Contact sheet
    print string
```

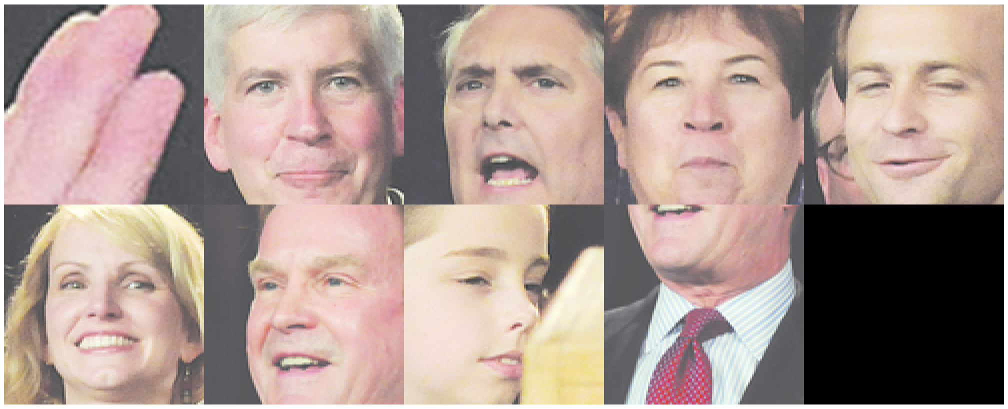
In [48]:
```
def resize_pic(img):
    baseheight = 100
    hpercent = (baseheight / float(img.size[1]))
    # With that ratio we can just get the appropriate height of the image.
    wsize = int((float(img.size[0]) * float(hpercent)))
    # Finally, lets resize the image. antialiasing is a specific way of resizing lines t
o try and make them
    # appear smooth
    img = img.resize((wsize, baseheight), Image.ANTIALIAS)
    return img
```

In [55]:
```
resize_pic(dict_file['a-0.png']['image'][4])
```
Out[55]:

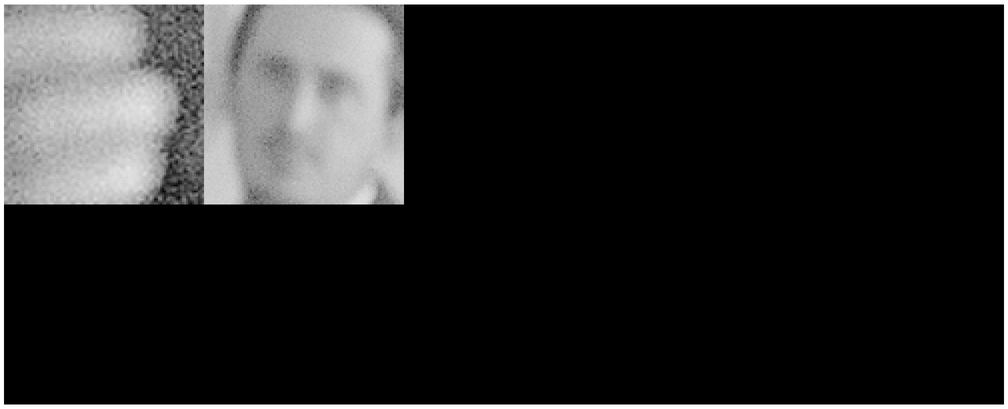
In [67]:
```
def search_key_word(search_word, dict_file):
    for i, j in dict_file.items():
        if search_word.upper() in j['text']:
            print("Results found in {}".format(i))
            if len(j['image'])>0:
                #print(len(j))
                first_image = resize_pic(j['image'][0])
                #display(first_image)
                contact_sheet=Image.new(first_image.mode, (500,200))

                x=0
                y=0

                for img in j['image']:
                    # Lets paste the current image into the contact sheet
                    img = resize_pic(img)
                    contact_sheet.paste(img, (x, y) )
                    if x+first_image.width == contact_sheet.width:
                        x=0
                        y=y+first_image.height
                    else:
                        x=x+first_image.width
                display(contact_sheet)
            else:
                print("But there were no faces in that file!")
```

```
In [81]: search_key_word('Christopher', dict_file)
```

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



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



Large file

```
In [96]: def parse_from_large_file(file_name_in_zip, zips, dict_file):
             img_extracted = zips.extract(file_name_in_zip)

             img_pil = Image.open(img_extracted)

             img = cv.imread(img_extracted)
             # And we'll convert it to grayscale using the cvtColor image
             gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

             faces = face_cascade.detectMultiScale(gray, 1.3) #1.3

             for x,y,w,h in faces:
                 dict_file[file_name_in_zip]['image'].append(img_pil.crop((x,y,x+w,y+h)))

             img_pil = img_pil.convert("L")
             # and run OCR
             text = pytesseract.image_to_string(img_pil)
             #print(text)
             for i in text.split('\n'):
                 if len(i)>3:
                     for j in i.split(' '):
                         dict_file[file_name_in_zip]['text'].append(j.strip().upper())

             return dict_file
```

In [97]:
```
large_zip = zipfile.ZipFile('readonly/images.zip')
file_names_in_large_zip = large_zip.namelist()
dict_file_large = defaultdict(defaultdict)

for file_name_in_zip in file_names_in_large_zip:
    dict_file_large[file_name_in_zip]['image'] = []
    dict_file_large[file_name_in_zip]['text'] = []
    dict_file_large = parse_from_large_file(file_name_in_zip, large_zip, dict_file_large
)
```

```
In [98]: search_key_word('Mark', dict_file_large)
```

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

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

```
Results found in a-10.png
```

```
Results found in a-2.png
```

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

```
Results found in a-5.png
```

```
Results found in a-8.png
But there were no faces in that file!
Results found in a-9.png
```

In [100]:
```
#dict_file_large
```

In [ ]:
```
```