# 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) of images and process them, using a library built into python 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), 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. 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 for testing.

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

If I were to use the 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!):

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 math
import zipfile
from PIL import Image, ImageOps, ImageDraw
import pytesseract
import cv2 as cv
import numpy as np

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

In [2]:

```
#define a parsed source to work on:
parsed_img_src = {}
```

In [3]:

```
#iterate through the zip file and save all the binarized versions to parsed_img_src
with zipfile.ZipFile('readonly/small_img.zip', 'r') as archive:
    for entry in archive.infolist():
        with archive.open(entry) as file:
            img = Image.open(file).convert('RGB')
            parsed_img_src[entry.filename] = {'pil_img':img}
```

In [4]:

```
#parse all images text
for img_name in parsed_img_src.keys():
    text = pytesseract.image_to_string(parsed_img_src[img_name]['pil_img'])
```

```
    parsed_img_src[img_name]['text'] = text
```

In [5]:

```
#find the bounding boxes for all the faces from every page and extract them
for img_name in parsed_img_src.keys():
    open_cv_image = np.array(parsed_img_src[img_name]['pil_img'])
    img_g = cv.cvtColor(open_cv_image, cv.COLOR_BGR2GRAY)
    faces_bounding_boxes = face_cascade.detectMultiScale(img_g, 1.3, 5)
    parsed_img_src[img_name]['faces'] = []
    for x,y,w,h in faces_bounding_boxes:
        face = parsed_img_src[img_name]['pil_img'].crop((x,y,x+w,y+h))
        parsed_img_src[img_name]['faces'].append(face)
```

In [6]:

```
#create thumbnails
for img_name in parsed_img_src.keys():
    for face in parsed_img_src[img_name]['faces']:
        face.thumbnail((100,100),Image.ANTIALIAS)
```

In [7]:

```
#search the keyword in every page's text and return the faces
def search(keyword):
    for img_name in parsed_img_src:
        if (keyword in parsed_img_src[img_name]['text']):
            if(len(parsed_img_src[img_name]['faces']) != 0):
                print("Result found in file {}".format(img_name))
                h = math.ceil(len(parsed_img_src[img_name]['faces'])/5)
                contact_sheet=Image.new('RGB',(500, 100*h))
                xc = 0
                yc = 0
                for img in parsed_img_src[img_name]['faces']:
                    contact_sheet.paste(img, (xc, yc))
                    if xc + 100 == contact_sheet.width:
                        xc = 0
                        yc += 100
                    else:
                        xc += 100

                display(contact_sheet)
            else:
                print("Result found in file {} \nBut there were no faces in that file\n\n".format(i
mg_name))
    return
```

In [8]:

```
search('Christopher')
```

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

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

In [9]:

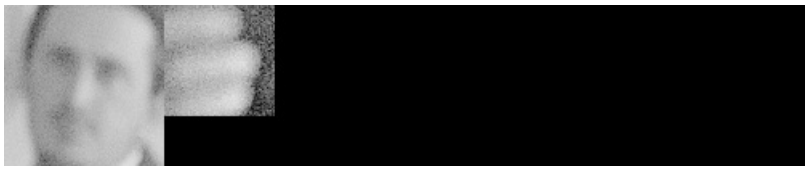
```
search('Mark')
```

Result found in file a-0.png

Result found in file a-1.png

Result found in file a-2.png

Result found in file a-3.png



In [10]:

```
search('pizza')
```

Result found in file a-2.png

In [ ]:

```

```