# 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:

Christopher Search

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!):

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 [8]:
```
#import zipfile
from zipfile 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')

# the rest is up to you!
```

In [9]:
```
# load files name and save them to a list
# extract files to unzip folder
def unzip(fname):
    output = []
    with ZipFile(fname, 'r') as z_name:
        name_lst = z_name.namelist()
        for name in name_lst:
            output.append(name)
    return output

unzip_lst = unzip('readonly/images.zip')
print(unzip_lst) #check if we have correct names from archive
```

```
['a-0.png', 'a-1.png', 'a-10.png', 'a-11.png', 'a-12.png', 'a-13.png', 'a-2.png', 'a-3.png', 'a-4.png', 'a-5.png', 'a-6.png', 'a-7.png', 'a-8.png', 'a-9.png']
```

In [11]:
```
#if we read archive, we will be returned binary-like objects so we need to convert the with io module
#found how to solve it here: https://stackoverflow.com/questions/32908639/open-pil-image-from-byte-file
#then load images and save them to a list, resizing didn't give any good effect so it's been removed
#preprocess images to grayscale then binarize them, preprocess to rgb for output at the end

def lst_of_img_gray(unzipped_lst):
    import io
    archive = ZipFile('readonly/images.zip', 'r')
    images_lst = [ Image.open(io.BytesIO(archive.read(i))).convert('L') for i in unzipped_lst ]
    return images_lst

def lst_of_img_rgb(unzipped_lst):
    import io
    archive = ZipFile('readonly/images.zip', 'r')
    images_lst = [ Image.open(io.BytesIO(archive.read(i))).convert('RGB') for i in unzipped_lst ]
    return images_lst

def binarize(image_to_process, threshold):
    image_processed = image_to_process.convert("L")
    for x in range(image_processed.width):
        for y in range(image_processed.height):
            if image_processed.getpixel((x,y))< threshold:
                image_processed.putpixel( (x,y), 0 )
            else:
                image_processed.putpixel( (x,y), 255 )
    return image_processed

def binarized_img(unzipped_lst):
    output = [ binarize(img, 256) for img in lst_of_img_gray(unzipped_lst) ]
    return output
```

In [12]:
```
#create a dictionary: every key is a file name, values are lists of words from pages (in lower case)
#NOTE: I didn't yet realize how to avoid word separation with '-'
def text_dict(unzipped_lst):
    temp_lst = []
    dict_of_words = {}
    count = 1
    img_lst = binarized_img(unzipped_lst)
    for img in img_lst:
        text = pytesseract.image_to_string(img)
        for word in text.split():
            if word not in temp_lst:
                temp_lst.append(word.lower())
        dict_of_words[unzipped_lst[count]] = temp_lst
        temp_lst = []
        count = count + 1
    return dict_of_words

dictionary = text_dict(unzip_lst) #
```

In [13]:
```
#function to find faces and create dictionary with faces collages as values and filenames as keys

def show_faces(unzipped_lst):
    from copy import deepcopy

    crop_dict = {}
    images_lst = lst_of_img_rgb(unzipped_lst)

    face_cascade = cv.CascadeClassifier('readonly/haarcascade_frontalface_default.xml')
    eye_cascade = cv.CascadeClassifier('readonly/haarcascade_eye.xml')
    archive = ZipFile('readonly/images.zip', 'r')

    for i in range(len(unzipped_lst)):
        crop_lst = []
        img = archive.read(unzipped_lst[i])
        img_decoded = cv.imdecode(np.frombuffer(img, np.uint8), 1) #load from buffer our image in bytes
        faces = face_cascade.detectMultiScale(img_decoded, 1.45) #took 1.45 scalefactor because it
        # detects less false positives
        pil_img = deepcopy(images_lst[i])

        contact_sheet = Image.new(images_lst[i].mode, (600, 300))

        for x,y,w,h in faces:
            crop_lst.append(images_lst[i].crop((x, y, x+w, y+h)))
        if len(crop_lst) != 0:
            x = 0
            y = 0
            for img_decoded in crop_lst:
                img_decoded = img_decoded.resize((100, 100), Image.ANTIALIAS)
                contact_sheet.paste(img_decoded, (x,y))
                if x + img_decoded.width == contact_sheet.width:
                    x = 0
                    y = y + img_decoded.height
                else:
                    x = x + img_decoded.width
        else:
            contact_sheet = "No faces are found in that "

        crop_dict[unzipped_lst[i]] = contact_sheet

    return crop_dict

crop_dict = show_faces(unzip_lst)
```

In [14]:
```
word_to_find = input('Enter a word to find: ')
```

```
Enter a word to find: Mark
```

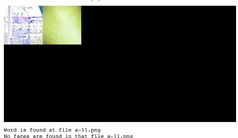
In [15]:
```
#logic for word search
#take enetered word, look in dictionary through keys for values which have entered word
#if word is found, then print results and pictures with faces
#if word is found but there are no faces, print results and 'No faces found...'
#if word isn't found no results are will be printed

for key in dictionary:
    if word_to_find.lower() in dictionary[key]:
        print('Word is found at file {}'.format(key))
        if type(crop_dict[key]) == str:
            print(crop_dict[key]+'file {}'.format(key))
        else:
            display(crop_dict[key])
```

```
Word is found at file a-0.png
```

```
Word is found at file a-1.png
```

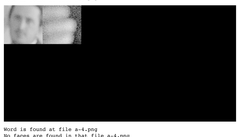
```
Word is found at file a-10.png
```



```
Word is found at file a-11.png
No faces are found in that file a-11.png
Word is found at file a-12.png
No faces are found in that file a-12.png
Word is found at file a-13.png
```

```
Word is found at file a-2.png
```

```
Word is found at file a-3.png
```

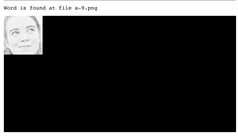


```
Word is found at file a-4.png
No faces are found in that file a-4.png
Word is found at file a-5.png
```


```
Word is found at file a-6.png
```

```
Word is found at file a-7.png
No faces are found in that file a-7.png
Word is found at file a-8.png
```

```
Word is found at file a-9.png
```



It took me 338 attempts to do that program for smal_img... Good project, it was hard) I will be glad to recieve comments to improve my programm