In [2]:
```
import PIL
import zipfile
import pytesseract
import cv2 as cv
import numpy as np
import kraken
from kraken import pageseg
from PIL import ImageDraw
from PIL import Image
# loading the face detection classifier
face_cascade = cv.CascadeClassifier('readonly/haarcascade_frontalface_default.
xml')
```

In [2]:
```
# Extarcting Images from Zip file
##=====================================================##
global img_names
zip_file = zipfile.ZipFile('readonly/images.zip')
img_names = zip_file.namelist()
zip_file.extractall('readonly/Untitled Folder')
##=====================================================##
```

In [3]:
```
# Importing Images
##============================================================##
images = []
for i in img_names:
    images.append(Image.open('readonly/Untitled Folder/' + i))
##============================================================##
```

In [4]:
```
print(images[0])
```

```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=3600x6300 at 0x7F769F82A
3C8>
```

In [5]:
```
global struct
struct = {}
for i in img_names:
    struct[i] = {}
```

In [6]:
```
text = []
# Extarcting Text from images
##=========================================##
text.append(pytesseract.image_to_string(images[0]))
##=========================================##
```

In [7]:
```
## Importing Images for Face Detection
##============================================================##
cv_images = []
for i in img_names:
    cv_images.append(cv.imread('readonly/Untitled Folder/' + i))
##============================================================##
```

In [8]:
```
count = 0
for i in img_names:
    struct[i]['PIL_img'] = images[count]
    struct[i]['cv_img'] = cv_images[count]
    count+= 1
```

In [9]:
```
count = 0
for i in img_names:
    struct[i]['text'] = pytesseract.image_to_string(images[count])
    count += 1
```

In [33]:

```
def detect_faces(img):
    ## Detecting Faces
    ##================================================================================
==============================##
    faces = face_cascade.detectMultiScale(struct[img]['cv_img'], scaleFactor=
2.0, minNeighbors=2, minSize=(10, 10))
    if faces == ():
        return "No face detected"
    ##================================================================================
==============================##
    ## Drawing Rectangles on the image
    ##=====================================================##
    img_copy = struct[img]['PIL_img'].copy()
    drawing=ImageDraw.Draw(img_copy)
    for x,y,w,h in faces:
        drawing.rectangle((x,y,x+w,y+h), outline="white")
    ##=====================================================##
    ##Cropping faces out of main image
    ##=====================================================##
    cr_img = struct[img]['PIL_img'].copy()
    cropped_faces = []
    for face in faces:
        x1 = face[0]
        x2 = face[0] + face[2]
        y1 = face[1]
        y2 = face[1] + face[3]
        cropped_faces.append(cr_img.crop((x1, y1, x2, y2)))
    ##=====================================================##
    ##Resizing cropped images to 100x100 Res.
    ##=====================================================##
    new = []
    lst = []
    for crop in cropped_faces:
        crop.thumbnail((100, 100))
    for i in cropped_faces:
        if i.size[0] and i.size[1] < 100:
            cropped_faces.remove(i)
    ##================================##
    # creating a contact sheet of faces detected
    ##================================================================================
================##
    if cropped_faces == []:
        return "No face detected"
    first_image=cropped_faces[0]
    contact_sheet=PIL.Image.new(first_image.mode, (first_image.width*5,first_i
mage.height*2))
    x=0
    y=0
    for img in cropped_faces:
    # Pasting the current image into the contact sheet
        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
```

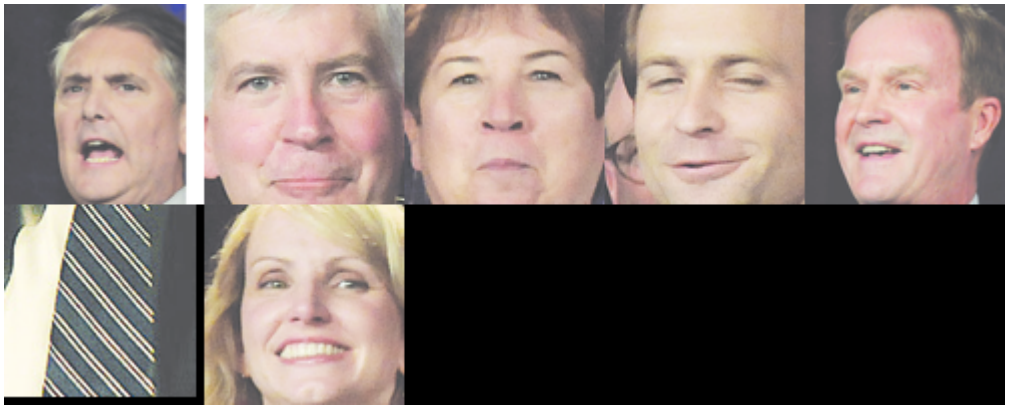
```
    # resize and display the contact sheet
    contact_sheet = contact_sheet.resize((int(contact_sheet.width),int(contact
_sheet.height) ))
    #=========================================================================
================##
    return contact_sheet
```

In [34]:

```
for i in img_names:
    struct[i]['cover'] = detect_faces(i)
```

In [35]:
```
user_in = input('Enter word to search:')
for i in img_names:
    if user_in in struct[i]['text']:
        print('Result found in file {}'. format(i))
        display(struct[i]['cover'])
```

Enter word to search:Mark
Result found in file a-0.png



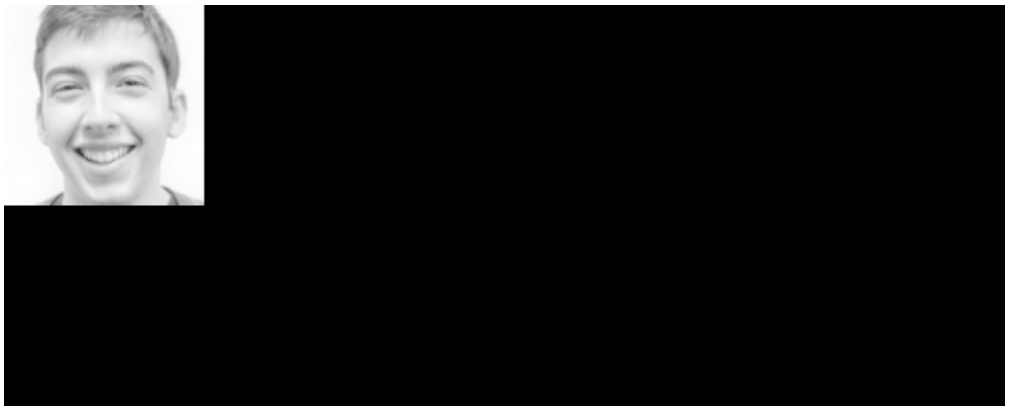
Result found in file a-1.png

Result found in file a-10.png

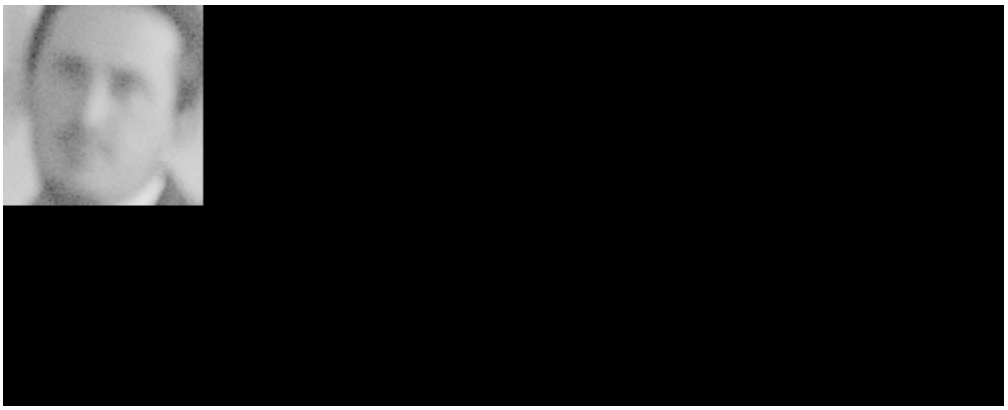
'No face detected'

Result found in file a-13.png

Result found in file a-2.png



Result found in file a-3.png



```
Result found in file a-8.png
```

In [ ]: