Add libraries

In [12]:

```
import cv2
import time
import numpy as np
```

Preparation for writing the ouput video

In [24]:

```
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))
```

Reading from the webcam(Use 0 if -1 doesn't work) 0 ,-1 represnts the webcam id

In [14]:

```
cap = cv2.VideoCapture(-1)
```

Allow the system to sleep for 5 seconds before the webcam starts. You can reduce this time. Linux takes a longer time than windows

In [15]:

```
time.sleep(5)
count = 0
background = 0
```

Capture the background in range of 60 for creating the background layer

In [16]:

```
for i in range(60):
    ret, background = cap.read()
background = np.flip(background, axis=1)
```

Color ranges you want to use:

In [17]:

```
# Red color
low_red = np.array([161, 155, 84])
high_red = np.array([179, 255, 255])
# Blue color
low_blue = np.array([94, 80, 2])
high_blue = np.array([126, 255, 255])
# Green color
low_green = np.array([25, 52, 72])
high_green = np.array([102, 255, 255])
# White color
low_white = np.array([0, 0, 0])
high_white = np.array([0, 0, 255])
#colors code

#skin color Values
#lower_red = np.array([0, 0, 70])
#upper_red = np.array([100, 255,255])
# mask1 = cv2.inRange(hsv, lower_red, upper_red)
```

Read every frame from the webcam, until the camera is open

In [18]:

```
while (cap.isOpened()):
    ret, img = cap.read()
    if not ret:
        break
    count += 1
    img = np.flip(img, axis=1)

    ## Convert the color space from BGR to HSV for capturing different intenseti
es of color
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    ## Generat masks to detect color

    ##YOU CAN CHANGE THE COLOR VALUE BELOW ACCORDING TO YOUR CLOTH COLOR


    # setting the lower and upper range for mask1
    lower_red = np.array([100, 40, 40])
    upper_red = np.array([100, 255, 255])
    mask1 = cv2.inRange(hsv, lower_red, upper_red)
    # setting the lower and upper range for mask2
    lower_red = np.array([155, 40, 40])
    upper_red = np.array([180, 255, 255])
    mask2 = cv2.inRange(hsv, lower_red, upper_red)

    mask1 = mask1 + mask2

    ## Open and Dilate the mask image
    mask1 = cv2.morphologyEx(mask1, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8))
    mask1 = cv2.morphologyEx(mask1, cv2.MORPH_DILATE, np.ones((3, 3), np.uint8))

    ## Create an inverted mask to segment out the red color from the frame
    mask2 = cv2.bitwise_not(mask1)

    ## Segment the red color part out of the frame using bitwise and with the in
verted mask
    res1 = cv2.bitwise_and(img, img, mask=mask2)

    ## Create image showing static background frame pixels only for the masked r
egion
    res2 = cv2.bitwise_and(background, background, mask=mask1)

    ## Generating the final output and writing
    finalOutput = cv2.addWeighted(res1, 1, res2, 1, 0)
    out.write(finalOutput)
    cv2.imshow("Bloody Muggle! Exit(q)", finalOutput)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

Save everthing and close windows (We put the release statements above so that the cell will move to idle state)