Submission Part 2

System Overview
The proposed solution utilises the opencv and numpy libraries. It uses opencv and mathematcal operations to detects features that were commonly observed in all images that exhibited lens flaring. All images to be analysed are converted into black-and-white binary images, with conversion parameters dependent on certain conditions, which are then run through specific functions which assess: the percentage of the screen covered with white pixels, the presence of clear white elliptical shapes, and presence of white rays of light.

Algorithm Outline and Intuition
The image is passed through 3 functions that check the aforementioned criteria; if it passes all 3 test a '0' is printed, otherwise if any of the tests fail, a '1' is printed and the program terminates. An 'if-else' ladder is utilised to ensure that the tests are run in a specific order, the last test (detection of rays) being the least reliable. Parameters pertaining to each function from the opencv library have been modified by trail and error to maximise correct classification of the data set given.

All functions are passed the original image since each test requires different tweaking of the parameters needed to convert it into a binary image. The images are first converted into a grey scale image and subsequently converted into the black-and-white binary image.

The first test is with *is_flare_lots()* function, which checks what percentage of the binary image consists of white pixels. The percentage cut off was determined empirically using the test set. Most cases should accurately assessed, except for images that have an ovecast sky at the center.

The second test is with the *is_flare_elliptical()* function. This test uses the opencv function *SimpleBlobDetector()* with parameters adjusted so that it detects ellipses. This is also another reliable test, its only weakness being sensitivity. If the ellipses aren't clear in the image then test will not detect the flare.

The last test is with the *is_flare_rays()* function. This test takes advantage of the opencv implementation of "Probabilistic Hough Line Transformation." The algorithm draws lines on the white section of the binary image and calculates their orientation as an angle. If there are white rays from flaring, then there will be sharp needle-like protrusions crossing the wall boundary in the image; the algorithm detects some of this (albeit unreliably). It was observed that if the Hough Line Transformation algorithim detects and draws lines ontop of these rays of light, the orientation will be between 30-60 degrees with respect to the horizontal. Thus if a line with an orientation between that interval is detected, the function returns a positive result.

Summary of Results
The program was able to accurately classify 28/40 of the flared images, and 34/40 of the good images in the dataset. The program incorrectly classifies good images as bad when the pictures are not perpendicular to the wall. There are also weaknesses with the ray detection function, where some cases fail to be noticed. Furthermore, there are cases where there are no ellipses or straight rays of light present in the image. An attempt was made to classify this scenario by processing the binary image to detect discontinuity between the sky-wall interface. This discontinuity resulted from the sun being directly in view of the camera, causing they binary image to have very roughly drawn arcs. Another type of Hough Line Transformation function was used to try and detect this, but I couldn't get it to work. The function for this type of detection can be seen in the source code *is_flare_arcs()*.

Submission Part 3

Preventative measures
Analysis with the proposed tagging algorithm on images recently taken should be used, where if the robot detects lens flaring, it should take pictures at different angles of the same area (if able). It is also possible, in the case where the sun is directly in line with the camera, to have something that is adjustable by the robot to cover the part of the lens that's directly taking in the sun; this should minimise lens flaring to some degree.

Algorithmic methods
After the image is detected, we could classify the type of lens flare distortion seen and treat each case appropriately. Lens flaring on the image creates out of place shapes with clear white boundaries: the circular bright spots (ellipses and circles), the rays of light, and the bright sun spots; all of these clearly visible on a binary image that is calibrated to highlight as much of the light as possible. The location of each of these can be isolated by approximating the true shape of the wall.

To isolate the true shape of the wall I would first of all find a way to remove or minimise the amount of trees above the wall. The true sky-wall boundary can then be approximated using a formula that takes into consideration the existing, clearly visible sky-wall boundary in the image. Super imposing the predicted boundary on the binary image representation of the original can be used to map the locations of flares. The flares can then be cut out, with the remaining empty space available for colouring in.

The empty space should be coloured with a method that reflects the type of flare. Clear round flares should be coloured in with an average of the true colour and pattern of the wall surrounding it. Rays should be coloured in with a gradient that reflects the changing wall pattern and colour around it, averaging the colour of the wall at the immediate sides of the ray is to its center. Images and sections with heavy flaring should be darkend.

The ideas above are my own, however I would also research existing methods and try to apply them.