**Summary Report**

In my assessment I covered project 4-1, which is the key point matching. I did this by using a brute force matcher with an ORB detector. I started the program by reading the two images to be compared and confirming that both images were valid, I created the ORB detector. This was where I encountered my first hurdle. Upon creating the ORB detector, the detector’s detectAndCompute function fails when trying to detect key points in template\_A.png because this image was mostly white space, the keypoint vector returned was empty and when passing that into the match() function, it crashes. This was due to the lack of key points found in the image so the detect function returns an empty keypoints vector which causes the program to crash when passed to the match function. I fixed this by changing the edge threshold of the detector from 31 to 15 so even key points were detected in that image. After creating the detector, I created the key points and descriptors and pass them both to the detect function with their corresponding image. This returns the key points and the descriptors for the image which I passed to the match function with a vector to contain the matches found. I then sorted the vector and took the first 15 matches as they were the good match, this reduced the amount of clutter found in the final image when I drew the matches. Which was where I encountered my second hurdle. Even after removing the bad matches, the outputted image contained a lot of colored circles for the key points which cluttered up the screen, so I added an extra parameter to not draw single points to the drawMatches function. This caused the function to only draw relevant matches, the lines connecting the two images and not draw the circles. I finished the project by resizing the final image to an appropriate size and displaying it.