- What did you learn after looking on our dataset?
  - It looks like the data is collected through different CCTV cameras (the naming convention c10, c20, c21 supports this hypothesis) etc in a parking lot. The CCTVs were on throughout the day probably, and images were captured after a regular defined interval. These have then been sorted according to time steps.

- How does your program work?
  - First, the code gets all the images in the data_path folder
  - Then creates a dictionary files_shapes to map files to their shapes while weeding out files that are corrupted.
  - Then, it creates a dictionary with keys as the camera id and list of files from that camera as values.
  - Then, it filters each list only keeping the images with the highest frequency of resolution per camera and sorts them.
  - Now, for each camera, it adds the first image to accepted files, then progressively checks the next image with the latest accepted file. If there is sufficient difference, it adds the current file to accepted files and then carries on with comparing further images to the new latest accepted file.
  - In the end, it returns a dictionary with key as camera id and values as the list of accepted files for the camera.
  - This list can be used to delete all files not present in the final dictionary

- What values did you use for input parameters and how did you find these values?
  - The score threshold is kept at 80,000. This number was found empirically as it returned unique images based on quantity and quality both. This was further refined using a manually selected 20 image subset to check how does the algorithm perform with various threshold values.

- What would you suggest to implement to improve data collection of unique cases in the future?
  - Firstly, the sampling frequency of capturing images should be reduced. For example, the state of a scene in a parking lot does not drastically change in 5 minutes. Hence, capturing images every 5 minutes will be redundant. For example, 30 minutes can be a good interval for capturing images.
  - A descriptor (for example SIFT features) can be used to store vectors of keypoints of images currently captured. Feature vector of a new image being captured should then be compared to the ones existing and this new image should only be stored if it is quite different than the images already existing in the database.
  - A deep learning based feature descriptor can also be used. Here, if two images are being compared for checking similarity, a siamese twin based deep learning approach can be used.

- Any further comments about your solution?
  - Yes, as the algorithm can only compare 2 images at a time, this task is essentially about reducing the number of comparisons to be done. I have used the most basic 2 pointer approach, as the dataset was tractable. There can be bigger datasets with millions or billions of images where a better approach can be used.