Answer to the given questions

1. **What did you learn after looking on our dataset?**
   The dataset appears to contain grayscale images of a parking lot, captured by different cameras and labeled with unique camera IDs and timestamps. Each image is associated with a camera ID. The camera ID helps identify the specific camera that captured the image. The timestamp indicates the time when the image was captured. It provides temporal information about the images. This information can be valuable for tasks such as tracking vehicles, analyzing occupancy, or detecting changes over time in the parking lot environment.

2. **How does you program work?**
   The program follows the following steps:

   - It starts by importing the necessary libraries for image processing and file operations.
   - It calls utility functions like draw_color_mask, preprocess_image_change_detection, and compare_frames_change_detection from the imaging_interview.py
   - The main part of the program starts by collecting a list of image files within the "dataset" directory using glob.glob.
   - Creates two folders, essentials_folder and nonessential_folder, if they don't exist.
   - Creates a copy of the dataset_folder named dataset_folder.copy.
   - It then iterates over each pair of image files in the dataset using nested loops.
   - For each pair, it loads and preprocesses the images using preprocess_image_change_detection.
   - The preprocessed images are then compared using compare_frames_change_detection, which calculates a score based on the difference between the two images.
   - If the score falls below a specified threshold (200,000 in this case), it implies that the images are similar, and the second image (file2) will move to the ‘non_essential’ folder.
   - After comparing all image pairs, it checks if any file was marked as a duplicate. If not, it moves the first file to the essentials_folder.
   - Logs the status of the file movements.
   - Closes the logger.

3. **What values did you decide to use for input parameters and how did you find these values?**
   In the provided code, the following values were chosen:

   Minimum contour area: 100 (the minimum area required for a contour to be considered significant).
   Score threshold: 200,000 (if the score is below this threshold, the second image is considered a duplicate).
   These parameter values are arbitrary and might not be universally optimal.

4. **What you would suggest to implement to improve data collection of unique cases in future?**

   To enhance the data collection of unique cases, you could consider the following:

   - Utilize camera ID and timestamp: Extract the camera ID and timestamp information from the filenames. Use this metadata to enhance the duplicate detection process.
   - Utilize hashing techniques: Compute image hashes (e.g., MD5, SHA-1, or perceptual hashes) and compare them to identify exact or near-duplicate images. This can provide a more precise and efficient way to identify duplicates.
   - Apply machine learning algorithms: Train a machine learning model to classify and identify duplicate images based on visual features. Deep learning models or image similarity algorithms could be employed for this purpose.

5. **Any other comments about your solution?**

   The provided code offers a basic implementation for detecting and removing duplicate images using simple change detection techniques. However, it may have limitations depending on factors such as image quality, variations, or the presence of noise in the dataset. It's important to thoroughly test and evaluate the code on the specific dataset to ensure its effectiveness and adjust the parameters accordingly. Additionally, consider incorporating the suggested improvements mentioned above to enhance the duplicate detection process.