Name: Cuong Ha; Matrik Nr: 03749410

Part 1: Skeleton Code
Inspect the next step function and describe a workflow for the implemented odometry method

The method processes on frames (each frame consists of an image pair from a stereo camera) in increasing order of appearance (i.e. timestamp).

- First, it selects the first frame as a keyframe.
- If a frame is selected as a keyframe, the following actions would be taken over its two images
    - Stereo matching to find inliers correspondences between them.
    - Project the current landmarks to the image plane using the current locations of the cameras
    - Find matches landmarks between the inliers in the current keyframe and current landmarks.
    - Localize the camera to update the keyframe's pose and set it as the current pose.
    - Add new landmarks from the left and right images.
    - Optimize the active map with bundle adjustment.
- If a frame is not selected as a keyframe:
    - Only find matches landmarks between the left image and current landmarks and localize the keyframe and update the current pose.
    - No stereo matching or optimizing or adding new landmarks.
    - A next frame would be chosen as a keyframe if the current number of inliers from localizing falls below a threshold.
- The process continues until it's done in all frames.

Part 3: Optimization
Difference between Optimize function in Odometry vs in Sfm:

- In Odometry, the optimize function only optimizes over cameras pose and landmarks 3D world frame location of current keyframes, in which the number of keyframes is always kept under a threshold. In Sfm, in the contrast, optimizing is done over all cameras and landmarks of the whole current map (which would be much slower with the growing number of frames). That means that optimizing in Odometry should be much faster than the counterpart of Sfm.
- In addition, in Odometry, bundle adjustment is done in a background thread

The functionality of the variables opt_finished and opt_running:

- The two variables are defined as atomic objects, in which accessing may establish inter-thread synchronization.
- The reason to use that is that optimizing current keyframes camera poses and landmarks with bundle adjustment is done in a background thread.
- The two variables provided flags for two actions:

    - Updating camera poses, landmarks, and calib_cam to the optimized version after bundle adjustment: only when optimization running is done (i.e. opt_running is false) and optimization is finished (opt_finished is true). After that, it set opt_finished to false to indicate that the process can take a new keyframe.
    - Taking a new keyframe: only if both opt_running and opt_finished are false.
- So the flags ensure the following order of actions with a background optimization process:
    - Optimizing is done in the background.
    - After optimizing finish, update the camera poses, landmarks, and calib_cam intrinsics.
    - After updating these optimized params, can add a new keyframe if needed.
- Without these flags, a new keyframe could be added before optimization over current keyframes is not finished;
    - Adding a new keyframe could trigger a new optimization process happening simultaneously with the current ones;
    - It could lead to conflict in results over camera poses and landmarks or accumulate errors because the first frame of new optimization is also fixed. And since it is not optimized by the current optimization, errors would be accumulated over time.