Task 1: Ring buffer

Following logic was used to create a ring buffer

```
If Vec.size() != FullCapacity:

   Vec.push_back(input);

else

   Vec.erase(Vec.begin());
   Vec.push_back(input);
```

Task 2: Keypoint detectors creation

For SIFT, BRISK, FAST, HARRIS Corner detector implementations from previous concepts in Tracking Image Features Lesson of Sensor Fusion Nanodegree were used.

For AKAZE, Following link was referred:
https://docs.opencv.org/3.4/d8/d30/classcv_1_1AKAZE.html

For ORB, Following link was referred:
https://docs.opencv.org/3.4/db/d95/classcv_1_1ORB.html

In general, logic following was quite uniformly applicable in all detectors implementation.

- Using detectorType string to control the flow of code
- Creation of detector object
- Creation of input parameters e.g. threshold, number of features if necerssary
- Using detect() method over the detector object

Task 3: Filtering Keypoints

Provided bounding box coordinates were used to filter out keypoints lying outside the area of interest.

Following logic was used for filtering:

- Created a vector car_keypoints for seperating the keypoints from ROI.
- Created an iterator to iterate over keypoints vector.
- Used contains() function over Keypoint vector indexed by its iterator to find if the keypoint lies in ROI
- If it lies in ROI then that keypoint was added to car_keypoints otherwise iterator was incremented.
- In the end, the car_keypoints vector were assigned to keypoints vector.

Task 4: Keypoint descriptors creations

For creation of SIFT descriptor the implementation from describe_keypoint.cpp was used.

For BRIEF, ORB, FREAK and AKAZE, following links were referred.

https://docs.opencv.org/3.4/d1/d93/classcv_1_1xfeatures2d_1_1BriefDescriptorExtractor.html

https://docs.opencv.org/3.4/db/d95/classcv_1_1ORB.html

https://docs.opencv.org/3.4/df/db4/classcv_1_1xfeatures2d_1_1FREAK.html

https://docs.opencv.org/3.4/d8/d30/classcv_1_1AKAZE.html

Task 5: Implementing FLANN matching and KNN selector

For FLANN based matching, following link was referred.
https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html

For KNN selector, above mentioned link was referred. As it gives some hints on how to implement knn matcher.

Task 6: Implementing distance Ratio test

I used following link for implementing distance ratio test
https://stackoverflow.com/questions/51197091/how-does-the-lowes-ratio-test-work

Following logic was formed

- Knn matches has 2 matches inside a vector.
- A loop was used to go over all knn matches and for each knn match, distances were substituted in distance ratio formula.
- If the condition of **if distance1 < distance2 * a_constant** held true for a match then the first keypoint among the 2 was added to the vector of goodKeypoints

Task 7, 8, 9: Comparison

For detailed stats of comparison, results.xlsx file can be referred. The summary of cimparison can be found below

Detector	Average Execution time in ms
FAST	7.54
ORB	17.98
HARRIS	24.07
SHI-TOMASI	69.91
AKAZE	70.02
SIFT	289

Descriptor	Average execution time in ms
BRIEF	6.19
ORB	25
BRISK	34.68
FREAK	36.12
SIFT	38
AKAZE	59

Detector-Descriptor pair	Matches per keypoints
FAST - ORB	0.79
FAST - BRIEF	0.71
FAST - BRISK	0.69
ORB- ORB	0.68
ORB - BRISK	0.70
BRB- BRIEF	0.51

Considering execution time of detector and descriptor at first and then considering the ratio of matches per keypoint, following detector descriptor pairs are recommended in given order.

Detector - Descriptor
FAST - BRIEF
FAST - ORB
FAST - BRISK
ORB - ORB