EE382N : Embedded System Design and Modeling
Lab #1
Names: Behzad Boroujerdian, Kishore Punniyamurthy, Keum San Chun

Problem (a)

1. Analyze the source code structure and draw a high-level block diagram of the function hierarchy and their communication dependencies (critical variables). **Submit the block diagram of the software architecture of the reference code as part of your lab report.** Note: in addition to the edge detection, the original source doe also supports image corner detection and smoothing. You can ignore functions that are not used in the edge detection code.

   Block diagram attached at the end of this report (Reference 1).

i) Critical variables were identified to be the following:
   in, mid, bp, r, x_size, y_size, bt, drawing_mode, max_no_corners, max_no_edges, mode

2. Modify the source code to be static and synthesizable. Modify the sources for a fixed input image sensor size of 76x95 pixels. Remove any unnecessary communication/dependencies. At the end, put each function called inside the main() function in a separate header/source file in order to make the example easier. **Report on the code changes that you performed.**

i) Modified block diagram attached at the end of the report (Reference 2).

ii) The functions that are only relevant to edge detection were retained.
   The followings are the changes made:
   1) in, mid, r were arrays of specific sizes. They were originally dynamically allocated by malloc(), we changed them to arrays of static sizes based on the input size.
   2) x_size and y_size were converted as constant parameters
   3) switch block was removed and replaced with a single case which corresponds to the edge detection.

3. Make sure that parameters passed between functions are not of pointer type.

   Actual arrays and constant values were used as parameters instead of pointers. The complete arrays were sent using port function calls of specC.

EE382N : Embedded System Design and Modeling
Lab #1
Names: Behzad Boroujerdian, Kishore Punniyamurthy, Keum San Chun

Problem (b)

1 & 2 & 3) Introduce a single behavior of appropriate name in each file. Let the behavior encapsulate all local variables and functions. Replace parameters with equivalent behavior ports for external communication. Make sure that parameters passed between functions are not of pointer type.

The following files were created:

1. stimulus.sc – this file contains the behavior which replaces the get_image() C function. It reads the image file and sends the image array to other behavior. It also sends a start signal to initiate the whole execution.

2. detect_edges.sc – this file contains the behavior setupbrightness and susanedges within it. The setupbrightness creates the array "bp" which is then passed to susanedges behavior using a queue. The susanedges behavior takes input from setupbrightness behavior and generates the "mid" and "r" arrays.

3. susan_thin.sc – this file contains the behavior which performs the susan_thin() C functions, it receives inputs from detect_edges and outputs "mid" to edge_draw through queue.

4. edge_draw.sc – This behavior corresponds to the edge_draw() C function. It receives the "mid' array from susan_thin and image from the stimulus behavior and outputs the result to monitor behavior.

5. monitor.sc – this behavior corresponds to the put_image() C function. It receives the image array from edge_draw through a queue and outputs it into a file.

6. main.sc – it is the behaviour where all the modules and ports/channels are instantiated. The channels instantiated are connected between appropriate behavior.

7. constant.hh - This files contains all the parameters required in other behaviors.

The top level intantiation:

```
c_double_handshake Trigger;

c_queue imageBuffer(img_size);

c_queue imageBuffer2(img_size);

c_queue detectEdgeOutputR(img_size4);

c_queue detectEdgeOutputMid (img_size);

c_queue susanThinOutput(img_size);

c_queue edgeDrawOutput(img_size);

stimulus myStimulus(imageBuffer, Trigger, imageBuffer2);
```

```
detectedges mydetectEdges(imageBuffer, Trigger, detectEdgeOutputR,
detectEdgeOutputMid);

susan_thin mySusanThin(detectEdgeOutputR,
detectEdgeOutputMid,susanThinOutput);

edge_draw myEdgeDraw(susanThinOutput, imageBuffer2, edgeDrawOutput);

monitor myMonitor(edgeDrawOutput);
```