EEL 4732/5733 Advanced Systems Programming
Assignment 5

In this assignment you are going to write a character special device driver by extending the simple driver we studied in class (see slides DDIntro.pptx) in the following ways:

1. **Define a device structure** and embed `structure cdev` in that structure:

   ```
   struct asp_mycdev {
      struct cdev dev;
      char *ramdisk;
      struct semaphore sem;
      int devNo;
      // any other field you may want to add
   };
   ```

2. **Make the major number configurable at load time.** Use 500 as the default value for the major number. Name this parameter as `majorno`.

3. **Support a variable number of devices** that can be set at load time (default will be 3) (see DDScullBasic.pptx). Name this parameter as `numdevices`. The device nodes will be named /dev/mycdev0, /dev/mycdev1, ..., /dev/mycdevN-1, where N is the number of devices. Please have the device driver create the device nodes.

4. **Make the size (in bytes) of the device buffer (ramdisk) configurable at load time** (see DDScullBasic.pptx). The default will be 16 pages as in the sample driver. Name this parameter `size`.

5. **Provide an entry function that would be accessed via `lseek()` function.** That entry function should update the file position pointer based on the offset requested. You should set the file position pointer as requested as long as it does not go out of bounds, i.e., $0 \leq requestedposition$. In the case of a request that goes beyond end of the buffer, your implementation needs to expand the buffer and fill the new region with zeros.

6. **Provide an entry function that would be accessed via `ioctl()` function (using the unlocked_ioctl field of file_operations struct).**

You should let the user application clear the data stored in the `ramdisk`. You should define symbol `ASP_CLEAR_BUF` for the command to clear the buffer. Your driver function should also reset the file position pointer to 0. Please see DDScullBasic.pptx for implementation details.

7. **Each device can be opened concurrently and therefore can be accessed for read, write, lseek, and ioctl concurrently.** It is your responsibility to provide appropriate synchronization to prevent race conditions.

8. All the resources (including the ramdisk, the major and minor nos, the device structures, and device nodes) should be **recycled/freed at the time of unloading your device driver module**.

Please submit all your files along with a Makefile and a README file on CANVAS.