Challenge1:

Make uEnv.txt to Boot from MMC0 and MMC 1
In this case, we loaded the boot images from MMC1 interface(eMMC) or MMC0 and used the file system present on the MMC0(MiroSD card) or MMC1, Challenge for you is to change this uEnv.txt such that boot images are loaded from MMC0(MicroSD) and MMC1 (eMMC).

Challenge 2:

Write a uEnvt.txt file to automate TFTP boot.
In this section you learnt how to do the TFTP boot and we are typing ***tftpboot*** commands each time we want to download the images. So, the challenge for you is to automate TFTP boot by writing uEnv.txt file . That means the boot images have to be downloaded automatically when you reset your board and should boot properly.
Also dont use the S2(boot) button during power up. The moment you press the reset button or give power to the board it should start fetching "dtb" , 'uImage", "initramfs" from TFTP.

Challenge 3:

Write a generic uEnv.txt
The challenge for you to write a generic uEnv.txt which can boot from any sources like eMMC , SD, TFTP, NFS, etc.
Use case : First it should try to boot from eMMC , if no boot images found , then it should move to SD, if no SD card found then it should try to get the images from TFTP , etc.

Challenge 4:

Increase the AUTOLOAD timings .
you might have observed that , when the uboot boots it just waits for 5 seconds before going to auto loading mode(reading uEnv.txt , loading uImage, etc). Within 5 seconds if you press "space" bar then you will get the uboot command prompt.
Challenge for you is to increase that autoload timing to 10 seconds and confirm this change by testing on the hardware. you have to change the uboot source code and then rebuild it to generate the uboot image.

Challenge 5:

Busybox "Dynamic" Compilation
In the entire "Busybox" lectures we have used "Static" binaries. That means all the generated utilities/commands of busybox are "statically linked" binaries. if you want to test any applications which are cross compiled by "Dynamic linking " then those applications wont execute on your Busybox file system.
The challenge for you is to reconfigure and re-compile the busybox to generate "dynamically linked " binaries/utilities and you should also able to test any applications which are dynamically linked.