# Report:

## Functionality of driver.cpp

To create this program, I first worked on the sthread\_yield() method of sthread.cpp. This is because the sthread voluntarily yields the CPU. Inside the method, I first check if alarmed is on. If it is, I first turn it off, then use setjmp() to memorize the thread context then I call the capture() function to capture it. The I used longjmp() for the scheduler. Then in the else for the setjmpm, I memcpy() the thread contents back to the stack. Inside the capture() method, I first get the stack pointer (sp) and base pointer (bp). Then I get the size of the stack by getting bp – sp. Then I set the cut\_tcp’s sp to the current sp. After that, some space is allocated in the head to save the activation record. Then I memcpy() current stack into the cur\_tcb’s stack. And lastly, I push the current thread into the queue.

Below are the illustrations of stack layers as well as execution snapshot.

### Chart (Illustration of the stack layers):

## A piece of paper with writing on it Description automatically generated

## My Output (Execution snapshot):

