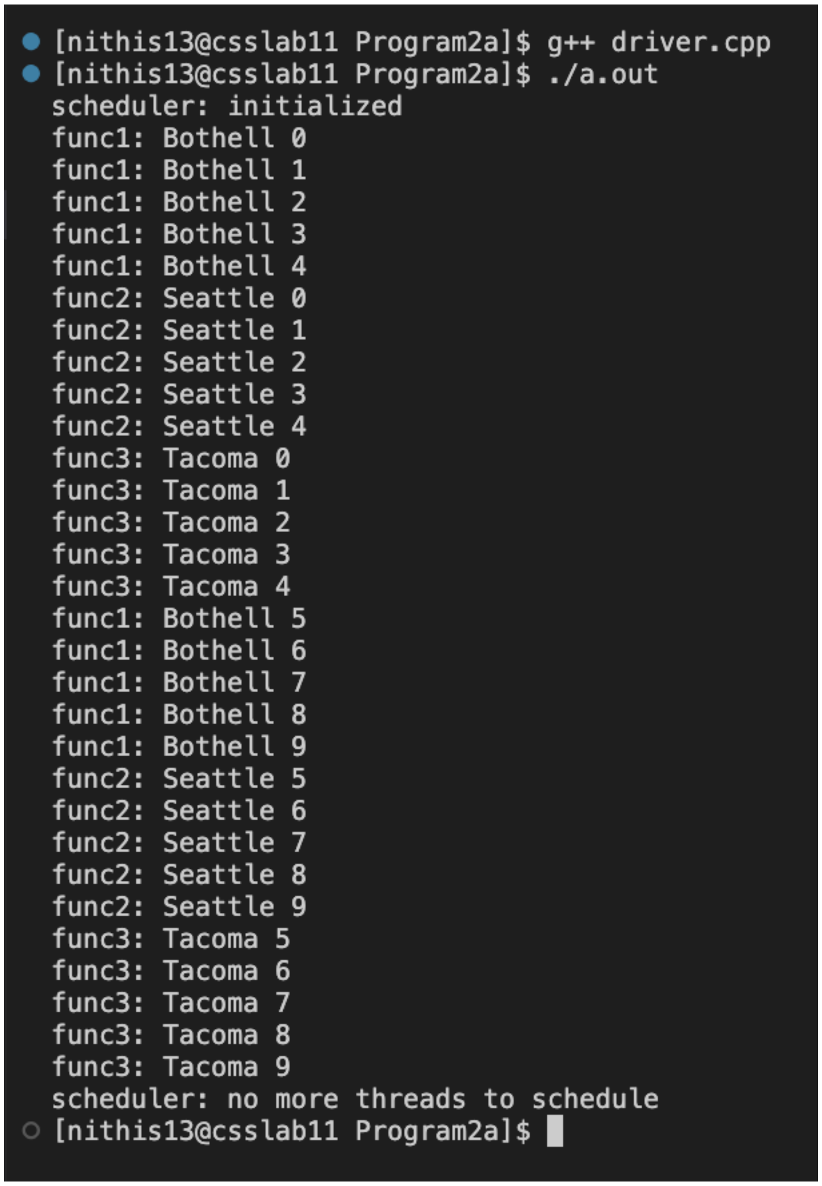
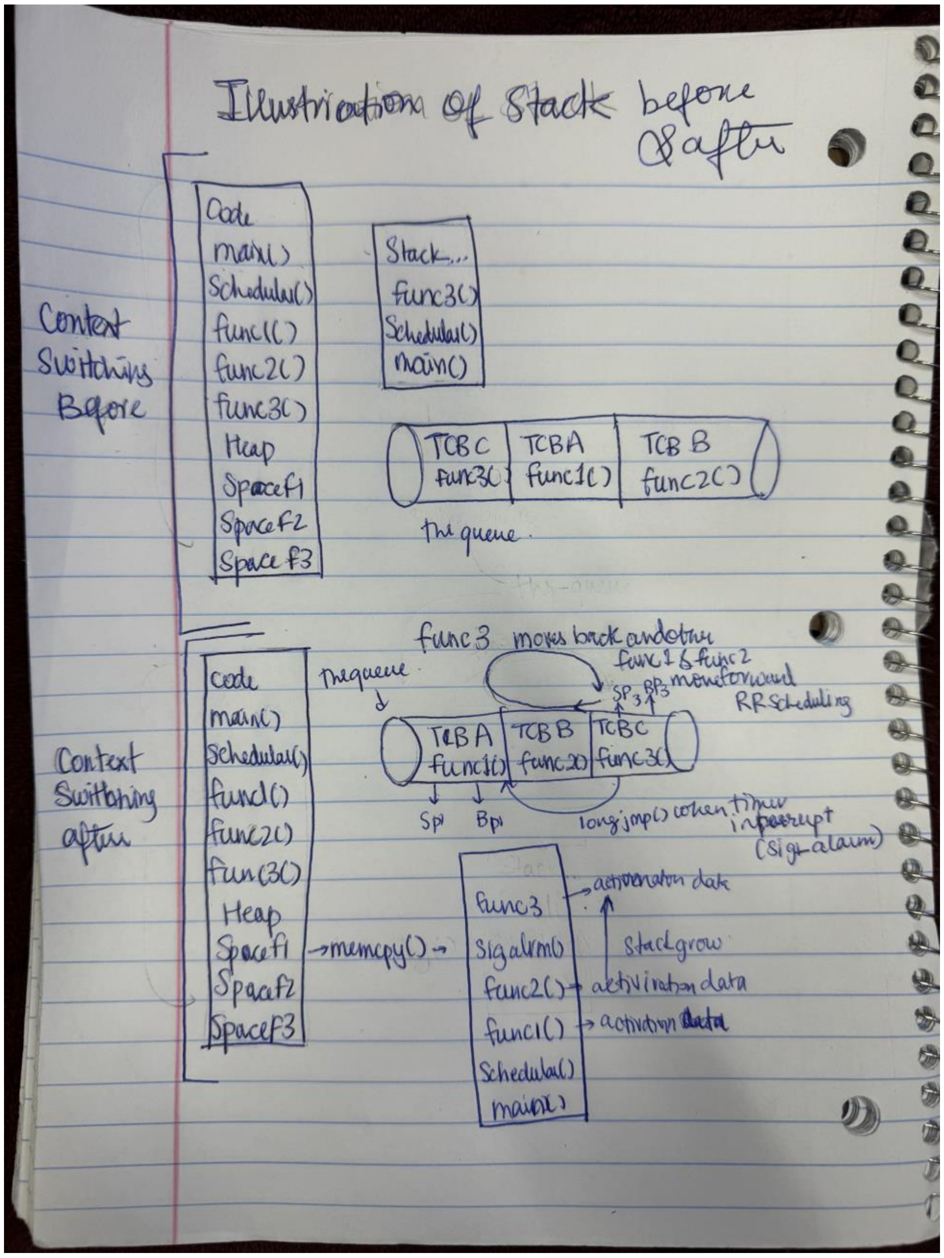
Program 2A Report

Screen Shots



```
● [nithis13@csslab11 Program2a]$ g++ driver.cpp
● [nithis13@csslab11 Program2a]$ ./a.out
  scheduler: initialized
  func1: Bothell 0
  func1: Bothell 1
  func1: Bothell 2
  func1: Bothell 3
  func1: Bothell 4
  func2: Seattle 0
  func2: Seattle 1
  func2: Seattle 2
  func2: Seattle 3
  func2: Seattle 4
  func3: Tacoma 0
  func3: Tacoma 1
  func3: Tacoma 2
  func3: Tacoma 3
  func3: Tacoma 4
  func1: Bothell 5
  func1: Bothell 6
  func1: Bothell 7
  func1: Bothell 8
  func1: Bothell 9
  func2: Seattle 5
  func2: Seattle 6
  func2: Seattle 7
  func2: Seattle 8
  func2: Seattle 9
  func3: Tacoma 5
  func3: Tacoma 6
  func3: Tacoma 7
  func3: Tacoma 8
  func3: Tacoma 9
  scheduler: no more threads to schedule
○ [nithis13@csslab11 Program2a]$
```

An illustration of the stack layers before and after



Each thread has its own stack, which is stored in TCB while it is not active. The scheduler controls these stacks using round-robin scheduling, which creates the illusion of concurrency. The essential context-switching techniques are implemented via capture(), sthread_yield(), and sthread_exit(), which save, restore, and free stack states respectively. The thr_queue guarantees that each thread receives a fair turn, with the round-robin technique performed by putting the active thread's TCB to the rear of the queue after yielding.