**Design and Algorithm**

This program implements thread context switching using preemptive round robin scheduling among three functions/threads: func1, func2, and func3. The program starts by running the functions in numerical order and only context switches when the alarm goes off indicating that the time quantum had ended. For this program, the quantum time has been set to 5 seconds. The 2 primary methods I have implemented are capture() and yield(). Capture works as a helper method to yield. Capture() is used to save the current threads context which consist of the stack pointer, base pointer, and stack. These three elements are saved into the cur\_tcb which is the thread control block. The Yield() method starts of by checking to see if the timer/alarm has gone off which indicates that the time quantum for the currently running thread has ended. If yes, then the timer is reset. Capture is then called to save the threads context and we longjmp to the schedular. The threads stack contents are then copied from the heap to the stack.

**Testing**

I tested my program fully on Linux. To see which parts of my code were being run at specific times, I used multiple cout statements. I also used multiple breakpoints and stepped through my program to see at which point in the program I get an exception/error.

**Output**

Correct RR context switch Incorrect RR context switch

A picture containing text

Description automatically generated Text

Description automatically generated with medium confidence

**Stack Layers when func3() is running**

A picture containing chart

Description automatically generated