**Program 2C Report**

This assignment is about round-robin schedulers and how context switch happens in a round-robin scheduler. In this assignment we learned how a round-robin scheduler works, which is having tasks or threads execute equal amount of time in turns until all the threads been fully executed [1]. We also learned how to implement the capture() function and the sthread\_yield() function using setjmp() and longjmp() system calls in part one of the assignment. The function capture() main purpose is to capture the functions, put it in a TCB, record the thread’s activation records, and put it into a queue for initiating. The sthread\_yield() function checks if an time interrupt has occurred and uses capture to capture the current thread if needed so we can yield to the CPU. In part two of the assignment, we illustrated a diagram which shows what happens when there is a context switch in a round-robin scheduler.

The output for part one of the assignment can be seen in figure 1.

A screenshot of a computer

Description automatically generated

Figure 1 – Output to round-robin scheduler

Once the scheduler is initialized, each function gets to run 5 seconds until a time interrupt occurs, which tells the scheduler to switch to the next function. You can see in the output that all functions are running 5 times (5 seconds) equally until all the function is done running. This shows exactly how a round-robin scheduler should behave.

The illustration for the context switch from function 3 to function 1 due to a time interrupt is depicted in Figure 2.

A diagram of a diagram

Description automatically generated with medium confidence

Figure 2 – Round Robin Context Switch

You can see that before the context witch, function 3 was still in the stack. Function 3 was also first in the queue as it is being executed. In the heap all the functions activation records are kept. After the context switch or time interrupt, function 3 gets moved to the end of the queue and we switch to the next function, which was function 1. You can see that we use memcpy() to copy function 1 activation record so we can continue to execute function 1 from where we left off.

**References**

[1] <https://www.guru99.com/round-robin-scheduling-example.html>