Jonathon Dahl
Georges Junior Naddaf

DECISIONS AND ASSUMPTIONS:

For task 1, we chose to implement our semaphore logic inside the ENTER_OPERATION and EXIT_OPERATION functions. This is simply to make the code clearer and for personal preference.
For task 2, we implemented alternating directions with which the buffer is traversed and implemented a merge_sort algorithm to boost code performance. Not using a sorting algorithm would have the code search for the highest/lowest value then the next highest/lowest value in the case of elevator algorithm making the code extremely inefficient. We could have used other sorting algorithms but merge_sort provides the best combination of runtime O(n log n) and ease of coding/debugging. We decided to use it and grabbed the code directly from http://www.geeksforgeeks.org/merge-sort-for-linked-list/ . As the sorting algorithm is not one of the main goals of this assignment, and after asking on piazza and unfortunately receiving no response, we assumed we could use it.
For task 3, we used simple clock_gettime operations to fetch the time. As the PA5 prompt did not specify a specific method to use, we went with that instead of signals and alarms.

PSEUDOCODE:

driver.c:

```
init();
sem_init(&buff_mutex, 0, 1);  // initialized as Lock
sem_init(&full_buffer, 0, 0); // initialized as CV

pthread_create(&disk_thread,NULL,disk_ops,algo);

void read_disk (int sector_number) {

        ENTER_OPERATION(t,sector_number); (Lock)
        read();
        temp1->req_id = sector_number;
        EXIT_OPERATION(t,sector_number); (Release) // wait for operation to finish up
        while(temp1->req_id != MAGIC);   // spin-lock for request to finish
}

void write_disk (int sector_number, int data) {

        ENTER_OPERATION(t,sector_number); (Lock)
        write();
        temp1->req_id = sector_number;
        EXIT_OPERATION(t,sector_number); (Release) // wait for operation to finish up
```

```
	while(temp1->req_id != MAGIC); //spin-lock for request to finish
}

void ENTER_OPERATION (char *op_name, int sector_number) {

	/* ENTER CS */
	sem_wait(&buff_mutex); // acquire the lock

	/* Add to buffer, wait if buffer is full - signaled from disk ops! */
	while(buff_count >= limit)
		sem_wait(&full_buffer);
}

void EXIT_OPERATION(char *op_name, int sector_number) {

	sem_post (&buff_mutex); // release the lock
	/* EXIT CS */
}
```

disk.c:

```
void *disk_ops(void *arg)
{


While (num_request_served < n) {

	clock_gettime(CLOCK_MONOTONIC, &beginWait); // begin time
	clock_gettime(CLOCK_MONOTONIC, &currentWait);

	While (buff_count < limit && (currentWait - beginWait) < TIME) {   //Spin lock + Timer

		sem_post(&full_buffer);		// signal that buffer empty
		clock_gettime(CLOCK_MONOTONIC, &currentWait);
	}



	if (*algo == 0); //do nothing, don't sort

	else if (*algo == 1) {

		// sort based on head direction
```

```
                MergeSort(&new_buffer_head);

                // Alternating directions to simulate elevator
                if(direction == 0)
                        direction = 1;
                else
                        direction = 0;
        }

        /*  In the case of the elevator algorithm, buffer traversal direction is being manipulated
        beforehand. The given FCFS algorithm works exactly like an elevator algorithm in this
        scenario. */

        simulation_algorithm();
    }
}
```

Jonathon Dahl
Georges Junior Naddaf

Average Service Times:

NOTE: buffer timer set to 5 seconds

Sample_input.txt		
	Average Disk Latency (s)	Average Service Time (s)
FCFS	0.2084443	0.4470291
Elevator	0.2090728	0.4439153
FCFS with buffer timer	0.2078879	0.4421133
Elevator with buffer timer	0.2073989	0.4403812

Sample_input_2.txt		
	Average Disk Latency (s)	Average Service Time (s)
FCFS	0.2095052	1.0883277
Elevator	0.2078467	1.0851105
FCFS with buffer timer (bug - negative times)	0.2086271	1.8431462
Elevator with buffer timer (bug - negative times)	0.2085136	1.8451873

Sample_input_3.txt		
	Average Disk Latency (s)	Average Service Time (s)
FCFS	0.2087527	1.6070352
Elevator	0.2081509	1.6051160
FCFS with buffer timer (bug - negative times)	0.2080264	2.4881310
Elevator with buffer timer	0.2085536	2.3604200

Jonathon Dahl
Georges Junior Naddaf

Sample_input_4.txt		
	Average Disk Latency (s)	Average Service Time (s)
FCFS	0.1990808	0.7721411
Elevator	0.2030198	0.9348065
FCFS with buffer timer	0.2070401	0.9379416
Elevator with buffer timer	0.2131695	0.8829188