Noah Boulton
V00803598

Design of P2

1. I will use 1 thread for each customer and 1 thread for each clerk as well as the main thread this means n customer threads + 2 clear threads + main thread = n + 3 threads total.
2. Threads work independently.
3. One mutex for each queue which guards the length of the queue, one mutex to guard the time, and one mutex for each clerk to allow them to suspend while waiting for customers.
4. The main thread will: Initialize mutex and condition variables
   Read customer information from customers.txt
   Create clerk threads
   Create the customer threads
   Wait for customers to terminate
   Destroy mutex and condition variables
   Calculate average waiting time for all customers
5. Customers will be a struct customer {
   int customer_id;
   int arrival_time;
   int service_time;
   double start;
   double end;
   int shortest;
   };
6. By using mutex locks to prevent concurrent access to global variables (queue lengths).
7. I use a convar for each queue and for each clerk
   Queue convars:
   a) A queue convar represents a clerk selecting a customer from the queue.
   b) The mutex for each queue will be associated with the convar to ensure that the queue is not being added to when the clerk is selecting from it.
   c) Figure out which clerk woke up the customer and print out its id
   Update the waiting time
   usleep() for the service time of that customer
   print out that service has ended
   pthread_cond_signal() the clerk so that it can serve another customer
   pthread_exit(NULL)
   return NULL
   Clerk convars:
   a) A clerk convar represents a clerk waiting for a customer to finish service.
   b) The mutex for each clerk will be associated with the convar to ensure that the clerk waits for the current customer to finish service before selecting another customer.
   c) Unlock the mutex lock and continue the next cycle
8. Main thread: Initialize mutex and condition variables
   Read customer information from customers.txt

Create clerk threads
Create the customer threads
Wait for customers to terminate
Destroy mutex and condition variables
Calculate average waiting time for all customers

Customer thread:
usleep() for the length of arrival time of this customer
Print that the customer arrives
select the shortest queue to enter
pthread_mutex_lock() the selected queue
print that the customer entered that queue
update the length of the selected queue
pthread_cond_wait(convar of selected queue, mutex of selected queue)
pthread_mutex_unlock(mutex of selected queue)
Figure out which clerk woke the customer up
Update the waiting time
usleep() for the service time of that customer
print out that service has ended
pthread_cond_signal() the clerk so that it can serve another customer
pthread_exit(NULL)
return NULL

Clerk thread:
while(true)
    check the length of the queues to see if there are customers waiting
    select the queue with the longest length
    pthread_mutex_lock(mutex of longest queue)
    pthread_cond_signal(convar of longest queue) wake the customer at the head of the queue
    pthread_mutex_unlock(mutex of longest queue)
    pthread_cond_wait() wait until the customer is done being served
pthead_exit(NULL)
return NULL