High-Level Description:

- Text file is read in from user.
- Customer events are organized into a multidimensional array with the first position being the customer, and the second being the data type.
- The Simulation class is instantiated.
- The first arrival is read into a fixed size (21) heap array at the first position. This is an array full of event objects, with each object containing: an arrival time, a primary service time, a secondary service time, an event time, an event type, and a server object. The event time is initially set to the arrival time of the customer, as the heap will be arranged by event times. The event type is initially set to 0, as 0 is an arrival event.
- The heap is sifted down using the siftdown algorithm.
- The main loop begins, while the event type of the customer at the top of the heap != 3:
- If the first item in the event queue (heap[0]) == 0:

i. The simulation handles it as an arrival event.

ii. The current time is set as the event time at the first position on the heap.

iii. If there is a primary server available:

1. Assign the available primary server to this customer, this server is now busy and starts working at the current time. The number of busy servers has now grown by 1.

2. The event time of the element of heap[n_busy] is set to the current time + the primary service time of that customer. This represents the position in the heap of the next idle primary server. The arrival time and secondary service time are also passed to this position in the heap.

3. This event is now given the event type of 1 so the main loop knows it's a primary service event.

4. This event is sifted up in the heap using the proper siftup heap algorithm.

iv. Else there are no primary servers available so we must enqueue the primary service time into the first queue data structure. The entire event object is actually copied and enqueued as the other data members are needed later but for now the primary service time is the most important. It is noted if the current queue size is its max.

v. When this is finished, the next arrival is read into the system and placed at the top of the heap, which removes the customer whose arrival was just handled. The heap is sifted down again.

vi. The while loop checking event type now starts again as the first event object at the top of the heap is now different.

• Else, if the event type at the top of the heap == 1:

. i) The simulation will handle a first service event.

. ii) The current time is set as the event time at the first position on the heap.

. iii) If there is a primary server available:

. (1) Assign the available primary server to this customer, this server is now busy and starts working at the current time.

. (2) The event time of the element of heap[n_busy] is set to the current time + the secondary service time of that customer. This represents the position in the heap of the next idle secondary server. The arrival time is also passed to this position in the heap.

. (3) This event type is now set to 2, as it is a secondary service event.

. (4) Once again we siftup

. iv) Else, there are no secondary servers available so we must enqueue the secondary service time into the secondary queue data structure. The other data members of the event are passed as well into the queue. It is noted if the current size of the second queue is its max.

. v) Now the primary service event at the top of the heap is handled by:

. vi) If the first queue is empty and has no waiting customers:

. (1) We set all the Event object elements in heap[0] to those of heap[n_busy] and we set all the elements of heap[n_busy] to 0. This represents the customer being served and taken out of the first line, completing their service.

. (2) The server who was serving the customer is now available, and switches (swaps) position in the array of primary servers to serve the next available customer or just be idle. The number of busy servers has now decreased by 1.

vii) Else, the first queue is not empty so the Event object elements at the top of the heap are set to those of the Event object elements at the front of the first queue, that customer in the queue is then dequeued.

viii)The heap is then sifted down.

- Else, if the event type at the top of the heap == 2:

. i) The simulation will handle a secondary service event.

. ii) The time is set to the event time at the top of heap.

. iii) If the second queue is empty:

. (1) We set all the Event object elements in heap[0] to those of heap[n_busy] and we set all the elements of heap[n_busy] to 0. This represents the customer being served and taken out of the second line, completing their service.

. (2) The server who was serving the customer is now available, and switches (swaps) position in the array of secondary servers to serve the next available customer or just be idle. The number of busy servers has now decreased by 1.

. iv) Else, the second queue is not empty so the Event object elements at the top of the heap are set to those of the Event object elements at the front of the second queue, that customer in the queue is then dequeued.

. v) The heap is then sifted down.

- Since there can only be up to 10 servers in each line and one possible arrival, there can only be, worst case, 21 events in the heap.

- When the last arrival is handled, instead of reading in a new arrival, the program sifts down an object with event time of 100,000, and an event type of 3. This holds the place of the possible arrival

event in the system to ensure that there can only be 10 servers of each kind working in the heap. When this event gets to the top of the heap, that means all other events have been handled and the simulation is over.

Data Structures Used:

. 1) Min Heap – a way of ensuring the next event in time would always be at the top of the array in logarithmic time without having to use something terrible like bubble sort.

. 2) Two Queues – one for each line, to hold the service times of the customers while there were no available servers.

. 3) Standard arrays.

. 4) Event and Server Objects.

Algorithms Used:

. i) Siftup – When creating the service time with the first idle server.

. ii) Siftdown – When reading in a new customer, or after a customer just completed their service.