1. **High‐level description of the overall solution strategy**

The C++ program initially reads the number of servers from the text file provided from the console. Then the program reads the first customer arrival time and service time from the file and store it into the event priority queue. Later inside the while loop, the program will extract a customer from event queue and check whether the event is a customer arrival or server finishing. If it is a customer arrival event, that customer is pushed into the customer queue and next customer arrival time and service time from the file is read and pushed into the event priority queue else if it is a server finishing event, a busy server from the heap of servers having least customer finishing time is extracted and pushed into heap of idle servers.

Now if a customer queue is not empty and there is an idle server available in the heap of idle servers, the program will extract a customer from the customer queue, assign to the most efficient idle server from heap of idle servers, push that server to the heap of busy servers and finally store that customer back to event priority queue with updated customer finish time. This whole process is done until every event is processed in the event priority queue. At the last following statistics is displayed on the screen: -

* Number of customers served
* Time last customer completed service
* Greatest length reached by the queue
* Average length of the queue
* Average customer waiting time
* Total number of customer served and idle time for each of the server

1. **List of all of the data structures used, where they are used and the reasons for their choice.**

**Linear arrays**

Where used: -

* Storing server elements.
* Storing event priority queue elements and customer queue elements.

Reasons: -

* Easy to store and access element in array.
* Both store and access takes constant time.
* Arrays take linear (O(n)) space in the number of elements n that they hold.

**Queues**

Where used: -

* Storing customer arrival time and customer service time using FIFO methodology.

Reasons: -

* Takes constant time for inserting an element
* Takes constant time for removing an element.

**Binary Heaps (Min Heap)**

Where used: -

* Storing event time, event service time and event type.
* Storing idle servers based on the server efficiency
* Storing busy servers based on the customer finish time.

Reasons: -

* Event can be inserted and extracted from the heap in O (log n) time.
* Server can be inserted and extracted from the heap in O (log n) time.

1. **List of standard algorithms used, where they are used and why they are used.**

**Binary Heaps (Min Heap): -** Priority queues can be efficiently implemented using Binary Heap because it supports insertion, deletion in O (log n) time.

Binary Heaps were used for implementing priority queues for storing events and servers.

This allows us to access the most efficient server in constant time. If the priority queue isn’t implemented in a heap, the program would have to do a linear search in the array of servers and find out the most efficient server.