CSCI203 - Assignment Two Report

Author: Dinh Che **Login:** dbac496 **Student ID:** 5721970 **Email:** dbac496@uowmail.edu.au

Overall Solution Strategy

1. Either a filename is passed as an argument to the program or a prompt is presented to ask for the filename.
2. The application opens the file and first reads in how many primary (`n_p_servers`) and secondary (`n_s_servers`) servers required.
3. The application opens the file for read and reads one line at a time which is converted to a `Customer` struct with `arrival_time` , `p_service_duration` , and `s_service_duration` values respectively read in.
4. The following data structures are initialised:
    1. `p_servers` and `s_servers` `Servers` objects which wrapper a 'first in first out' queue `Server` struct array based on how many are required.
    2. `p_server_q` and `s_server_q` `ServerQueue` objects which is a wrapper for a 'first in first out' queue implemented with `Customer` struct array.
    3. `event_q` which is an `EventQueue` wrapper object for a priority queue implemented as a minimum value heap based on event time.
5. The first event is read in from the file and added to the `event_q`
6. The next event from the file is read based on a `cust_arrival_flag` to check whether there are anymore events in the file to read.
7. The first event is extracted from the `event_q` and processed as the following cases:
    1. `eCustomerArrived` enum:
        1. customer is served by a p server if available, and event_time is set to when service complete
        2. customer is put in a p server queue if no server available, time they went in queue stored
    2. `eCustpFinished` enum (customer has finished with p server):
        1. customer is served by a s server if available, and event_time is set to when service completed
        2. customer is put in a s server FIFO queue if no server available, time they went in queue stored
    3. `eCustSecondaryFinished` enum
        1. customer has finished with a s server and being served
        2. statistics are calculated
8. If there are `Customers` waiting in the `p_server_q` or `s_server_q` , and there are servers available, the customers is extracted from the head of the queue and processed.
9. Statistics are calculated and printed.

Data Structures Used

- `Customer` struct to store some key information about each customer such as:

    - `arrival_time` - Arrival time to the shop.
    - `p_service_duration` - How long the customer spends with a primary server.
    - `s_service_duration` - How long the customer spends with a secondary server.
    - `wait_duration` - How long the customer has waited in a primary or secondary server queue.
    - `cust_queued_time` - Total time the customer has waited in queues.
- `Event` struct to store key information about each event to process:
    - `type` - Type of event from the enum (`EventType { eCustomerArrived, eCustPrimaryFinished, eCustSecondaryFinished }`)
    - `ev_time` - The discrete time during the simulation the event must be processed.
    - `cust` - The customer relevant to each event.
- `Server` struct to store key information about the server such as statistics and their index in the array.
- `EventQueue` class which is a wrapper class to encapsulate class functions relevant to the data structure.
    - A priority queue implemented as a heap data structure which has at worst $O(\log n)$ insert and $O(1)$ to extrac the next event.
- `Servers` class which is a wrapper class to encapsulate class functions relevant to the data strucutre.
    - An array `_servers` to store the `Server` struct based on their index.
    - An array `_n_idle_servers` which is a 'First In First Out' queue using index numbers of the `_servers` array to efficiently find the next idle server.
    - Initialising the servers costs $O(n)$, to enqueue the server in the `_n_idle_servers` array is $O(1)$ and to dequeue the next available server is $O(1)$.
- `ServerQueue` class which is a wrapper class to encapsulate class functions relevant to the queue.
    - An array `_serv_q` which stores `Customer` structs in a 'First In First Out' queue implemented as an array with a head and tail index.
        - Enqueue for this implementation is $O(1)$ and dequeue is also $O(1)$

Standard Algorithms Used

- Nil