Understanding the Similarities (and Differences) between the "Airport Shuttle Bus (ASB)" system and the "Gas Station (GS)" system

To simplify the task of converting the "gas station (GS)" code in C++ to a CSIM code, this document helps you to understand the similarities between the ASB system and the GS system, so that you can modify the ASB code (or follow the structure of that code to write new CSIM code) for this lab and future CSIM coding assignments. As we step through the ASB code, you will also be introduced to the generic principles of modeling a system with CSIM objects.

To start, notice that in the ASB model, there are two streams of arriving customers: one at the airport terminal and the other at the rental car lot. The two queues were representative of two separate single queue single server processes, and therefore modeled as two "facilities" in CSIM.

Compare this to the "gas station (GS)" code. In the GS system, there is a single queue of arriving cars, but there are N pumps that may serve the cars. This represents the case of single queue, multiple server process and can therefore be conveniently modeled as a "multiserver facility" in CSIM. (Notice that the gas station can alternately be modeled as a single storage with a capacity of N, where the pumps are allocated and deallocated as needed).

If we remove one of the facilities from the ASB code, say the rental car lot, and the associated queue, then the ASB code would model only a single stream of arriving customers at the terminal, who join the queue at the terminal. If we throw away the interactions of the customers with the shuttle bus, then, other than the arrival process, the ASB code would only have one more process, which is the arr_cust () process. This process defines the activities of a customer at the terminal from the time that it arrives (is born), until the time that it departs the terminal facility (dies).

The sequence of actions that are associated with a customer in the car lot are: arrive (be "born"), try to reserve a resource (push call button), do something that takes time (board the shuttle), release the resource (release the call button) and depart ("die").

Therefore, the inclusion of the following processes can completely model the ASB system with only one stream of arriving customers: main process, arrival of customers at car lot, activities of a customer at car lot, shuttle.

Now consider the gas station (GS) code. If we restrict N (number of pumps) to 1, then the gas station is no longer a multiserver facility, rather it becomes a single server facility with a single queue, just like one of the facilities in the ASB code. If we remove all parts of the ASB code pertaining to one of the two streams of arriving customers, then the GS code will be similar to the remaining ASB code. The cars would then act just like the car lot customers: arrive (be "born"), try to reserve a resource (use a pump), do something that takes time (pump gas), release the resource (release pump) and depart ("die").

The GS code can be modeled with the following processes: main process, arrival of cars at the gas station, activities of a car at the gas station.

At the start of the ASB CSIM code, the constants are defined. The constants for the ASB code are: number of seats in the shuttle, time taken by a single customer to board shuttle. Similarly at the start of the CSIM version of the GS code, the relevant constants should be defined. One of the constants is profit per liter sold. You may define more constants depending on how you design your code.

Next, in the ASB code, are the declarations for elements of the system to be modeled - each of which is an appropriate CSIM object: facilities, events and processes. (The prototypes for ordinary functions are also declared at this point. Note the difference between processes and ordinary functions in the code). Similarly in the GS code, you must declare a suitable CSIM object for all the elements in the model. Also declare the processes and the prototypes for any functions that you will use in the code.

Once the elements of the system have been declared, the main process "sim" is created. The structure of the main process in any CSIM code that you will write will generally be as follows:

```
extern "C" void sim()
{
        create("sim");
        // start simulation by forking the arrival process and other processes that need to be forked here
        // wait until the entire simulation time elapses, taking snapshots as required
        // print report (required statistics as per problem definition)
}
```

Notice that the first statement within a process is always a "create" statement.

In the ASB code, the next process that is defined is the process that models the arrival of the customers, which is arrivals(). For the ASB system, the customers are assumed to arrive in groups, with exponential inter-arrival times (poisson process). For the GS code, the only differences are: the cars arrive one at a time instead of arriving in groups, and the mean inter-arrival time is different.

After the arrival process, the arr_cust() process in the ASB code defines the activities of a customer in the terminal: after arrival, customer joins the queue, controls a resource (pushes call button), spends some time (waits), releases the resource (releases call button), and departs. For the GS code, we would similarly have a process, say car(), which defines the activities of a car: after arrival, the car makes a decision to stay or balk, depending on pump availability, it either joins the queue or immediately starts pumping gas, controls a resource (reserves pump), spends some time (pumps gas), releases the resource (releases pump), and departs. Notice that since the GS code requires us to keep some statistics about the arriving cars (stays or balks?), queue lengths, liters pumped etc., we will also need to call within the process, appropriate functions of relevant objects (facilities, meters, etc.) to maintain the required statistics.

Other than the processes, the ASB code has one helper function group_size(). While modeling the gas station, you can have one or more helper functions in the CSIM version of the GS code, depending on how you choose to design your code. For example, one helper function maybe periodic_report(), which helps you to compute and display statistics during the periodic snapshots. Note that you can gather many statistics by simply calling inspector functions of the predefined CSIM objects, which allow you to retrieve attributes of the objects. In particular, standard functions to retrieve queue length etc. are already provided. For a complete list of all inspector functions provided for a particular CSIM object, see the user guide. You must learn to use them effectively to leverage the functionalities that CSIM provides, and simplify your code.