Birla Institute of Technology and Science, Pilani Hyd Campus

CS F211: Data Structures and Algorithms

2nd Semester 2024-25 Lab No:7 (Queue & Deque ADT)

Program 1: You are given a sequence of characters stored in a deque. Your task is to determine whether the characters in the deque can be rearranged to form a palindrome. At any given moment, you can pick a character from either end of the deque. This process continues until the deque is empty, after which the characters that you have removed are arranged in the order in which they were picked and checked for palindrome. If a path fails, you need to backtrack and restore the characters in the deque to try other possibilities. Write a recursive function to do the same.

Two test cases are as shown below. Prog1.cpp (attached is the incomplete code) has a code skeleton. You need to write the complete code using the logic discussed in the class.

```
Enter the number of characters in deque: 5
Enter the characters: a b c d e
NO
```

```
Enter the number of characters in deque: 6
Enter the characters: a a b b c c
YES
```

Program 2: (C++ STLs allowed) Supermarket Checkout Simulation

In this program you will apply queue concepts learnt in the class to a complex real-world scenario by simulating a supermarket checkout system. Simulate the Supermarket's (like Agarwal supermarket in the campus) checkout process with multiple checkout lanes. Customers arrive at the supermarket at random intervals and choose the shortest available checkout lane after completing their purchases. Each checkout lane (like you have two checkout lanes in the campus Supermarket) has a queue where customers wait for their turn. Cashiers serve customers at the front of their respective queues.

You may use the below simulation parameters:

num_lanes: The number of checkout lanes (e.g., 2); arrival_rate: The average number of customers arriving per unit time (e.g., customers per minute, use a random number generator to simulate inter-arrival times between customers); service_time: The average time it takes for a cashier to serve a customer (e.g., minutes per customer, you can assume a fixed service time or make it random); simulation_time: The total time to simulate (e.g., minutes).

You should use the following data structures or ADTs:

1. You must implement a 'Customer' class in C++ to store customer information, at minimum their arrival time.
2. You must implement a 'CheckoutLane' class to manage each checkout lane's queue and cashier. You can use 'std::queue' (STL) for the queue within this class.

Below are the required functions/ classes that you may use:

```
class Customer{
public:
      Customer(int arrival_time);// Constructor
      int getArrivalTime() const;// Returns the arrival time
private:
      int arrival_time;
};
class CheckoutLane{
public:
      CheckoutLane(); // Constructor
      void enqueue(const Customer& customer); // Adds a customer
      Customer dequeue(); // Removes and returns the customer
      bool isEmpty() const; // Returns true if the queue is empty
      int queueLength() const; // Returns the number of customers
      bool isCashierAvailable() const; //Returns true if the cashier is available.
      void setCashierAvailable(bool available); // Sets the cashier's availability.
      int getCustomersServed() const;//Returns the number of customers served by this lane.
      void incrementCustomersServed(); // Increments the number of customers served.
      Customer getCurrentCustomer() const; // Returns the customer currently being served.
      void setCurrentCustomer(const Customer& customer);//Sets current cust. being served.
private:
      std::queue<Customer> queue;
      bool cashier_available;
      int customers_served;
      Customer current_customer;
};
void supermarket_simulation(int num_lanes, double arrival_rate, int service_time, int simulation_time){

      // This function performs the entire supermarket simulation. Write your code here…

}
```

```
// For generating random interarrival times (exponential distribution). You might need these:
#include <random>
#include <chrono>
double generate_random_interarrival_time(double arrival_rate){
        static std::random_device rd; // Obtain a random seed from the OS
        static std::mt19937 gen(rd());// Standard mersenne_twister_engine seeded with rd()
        std::exponential_distribution<>distrib(arrival_rate);//Arrival rate is mean for exp
return distrib(gen); // Generate a random number from this distribution
}
// For generating random service times (you might want a different distribution)
int generate_random_service_time(int service_time){
        static std::random_device rd;
        static std::mt19937 gen(rd());
        // Example: uniform distribution around the average service time
        std::uniform_int_distribution<> distrib(service_time*0.5, service_time*1.5);
 return distrib(gen);
 }
```

Sample Testcases:

```
Customer arrived at time 2 and joined lane 1
Customer served at lane 1 at time 2 with service time 5
Customer arrived at time 9 and joined lane 1
Customer arrived at time 9 and joined lane 2
Customer served at lane 1 at time 9 with service time 5
Customer served at lane 2 at time 14 with service time 5
Customer arrived at time 24 and joined lane 1
Customer served at lane 1 at time 24 with service time 6
Customer arrived at time 32 and joined lane 1
Customer served at lane 1 at time 32 with service time 7
Customer arrived at time 46 and joined lane 1
Customer served at lane 1 at time 46 with service time 3
Customer arrived at time 60 and joined lane 1
Customer served at lane 1 at time 60 with service time 3
Customer arrived at time 65 and joined lane 1
Customer served at lane 1 at time 65 with service time 7

--- Simulation Summary ---
Simulation ended at time: 74 minutes.
Lane 1 served 7 customers.
Lane 2 served 1 customers.
Total customers served: 8
```

```
Customer arrived at time 2 and joined lane 1
Customer served at lane 1 at time 2 with service time 3
Customer arrived at time 9 and joined lane 1
Customer arrived at time 9 and joined lane 2
Customer served at lane 1 at time 9 with service time 4
Customer served at lane 2 at time 13 with service time 3
Customer arrived at time 22 and joined lane 1
Customer arrived at time 22 and joined lane 2
Customer served at lane 1 at time 22 with service time 6
Customer served at lane 2 at time 28 with service time 4
Customer arrived at time 36 and joined lane 1
Customer served at lane 1 at time 36 with service time 7
Customer arrived at time 47 and joined lane 1
Customer served at lane 1 at time 47 with service time 7
Customer arrived at time 55 and joined lane 1
Customer arrived at time 55 and joined lane 2
Customer served at lane 1 at time 55 with service time 7
Customer served at lane 2 at time 62 with service time 3

--- Simulation Summary ---
Simulation ended at time: 67 minutes.
Lane 1 served 6 customers.
Lane 2 served 3 customers.
Total customers served: 9
```

Deliverables: C++ code and output snippets with multiple test cases to be uploaded to the class page at the end of the lab.

Program 3: Prog3.cpp the implementation of a queue using Circular linked list as discussed in the class. This file is attached alongwith the lab sheet for you to carry out a sample run to get the below output.

```
1 : Enqueue
2 : Dequeue
3 : Get front element
4 : Get size
5 : Check if queue is empty
6 : Display Queue items
7 : Exit
+-------------------------+
Enter your choice: 1
Enter an item: Pilani
Pilani enqueued into the Queue.
+-------------------------+
Enter your choice: 1
Enter an item: Dubai
Dubai enqueued into the Queue.
+-------------------------+
```

```
Enter your choice: 1
Enter an item: Goa
Goa enqueued into the Queue.
+-------------------------+
Enter your choice: 1
Enter an item: Hyd
Hyd enqueued into the Queue.
+-------------------------+
Enter your choice: 6
FRONT -> Pilani Dubai Goa Hyd
+-------------------------+
Enter your choice: 2
Pilani dequeued from the Queue.
+-------------------------+
Enter your choice: 3
Elem at the FRONT = Dubai.
+-------------------------+
Enter your choice:
```

Now your task is to modify the same code (Prog3.cpp) to get the below output for the same input that you gave in the upper part, i.e. Pilani, Dubai, Goa and Hyd. (You are free to modify any function i.e. enqueue, dequeue, traverse, add, remove etc.).

Output:

```
Enter your choice: 6
FRONT -> Dubai Goa Hyd
+-------------------------+
Enter your choice:
```
