
DS2030 Data Structures and Algorithms for Data Science

Lab 3

August 26th, 2025

Lab Instructions

- Create a folder named “**DS2030_<RollNo.>**” (all letters in capital) in “**home**” directory.
Eg- **DS2030_142402022**
- Name the script files in the given format
“**<your_roll_no>_<Name>_Lab3.py**”
- Make sure the **folder, files, classes, functions and attributes** are named as instructed in the lab sheet.
- We will not be able to evaluate your work if the folder is not properly named or is not located in the home directory.
- Make sure to save your progress before leaving the lab.
- Do not shut down the system after completing the exam.
- You are not allowed to share code with your classmates nor allowed to use code from the internet. If we find that you have copied code from your classmate or from the Internet, you will get a straight fail grade in the course.

Ticket Queue Management System

Problem Statement: Imagine a ticketing counter at a theater, bank, or any customer service desk. Customers arrive and take tickets to wait for their turn. The queue has several key operations:

- Customers arrive: They join the end of the queue.
- Customers are served: The one at the front leaves the queue.
- Peek at the next customer: Staff can see who will be served next without removing them.
- Cancel a booking: A customer may leave the queue before being served.
- Move to end: A customer can give up their current position and go to the back.
- Find position: Staff may check the position of a customer in the queue.
- Count total customers: Staff can check how many customers are still waiting.

Data Structure: A linked list is used where:

- Each node represents a customer with `customer_id` and `customer_name`.
- The queue maintains pointers to the front and rear nodes.
- `total_customers` keeps track of the number of customers in the queue.

Functions:

- **enqueue**: Adds a new customer to the rear of the queue.
- **dequeue**: Removes the customer at the front of the queue.
- **peek_next_customer**: Shows who will be served next without removing them.
- **cancel_booking**: Removes a specific customer by ID.
- **move_to_end**: Moves a customer to the end if they choose.
- **find_position**: Checks the position of a customer in the queue.

Starter Code:

```
1 class Node:
2     def __init__(self, customer_id, customer_name):
3         """
4             Initialize a node for a customer.
5             - customer_id: Unique ID of the customer.
6             - customer_name: Name of the customer.
7             - next_node: Pointer to the next node in the queue.
8         """
9         # TO DO: initialize customer_id, customer_name, and next_node
10
11
12 class TicketQueue:
13     def __init__(self):
14         """
15             Initialize an empty ticket queue.
16             - front_node: Points to the first customer in the queue.
17             - rear_node: Points to the last customer in the queue.
18             - total_customers: Number of customers in the queue.
19         """
20         # TO DO: initialize front_node, rear_node, and total_customers
21
22     def enqueue(self, customer_id, customer_name):
23         """
24             Add a customer to the end of the queue.
25             - Update front and rear pointers.
26             - Increment total_customers.
27             - Return customer_name.
28         """
29         # TO DO: implement enqueue logic
30
31     def dequeue(self):
32         """
33             Remove and return the first customer.
34             - Return None if queue is empty.
35             - Otherwise, remove first customer and update front/rear if needed.
36             - Decrement total_customers.
37             - Return customer_name.
38         """
39         # TO DO: implement dequeue logic
40
41     def peek_next_customer(self):
42         """
43             Return the name of the next customer without removing.
44             - Check if queue has at least two customers.
45             - If not, return None.
46             - Otherwise, return the next customer's name.
47         """
48         # TO DO: implement peek logic
49
```

```

50     def cancel_booking(self, customer_id):
51         """
52             Remove a customer using their ID.
53             - Return customer_name if removed.
54             - Update front/rear if needed.
55             - Decrement total_customers.
56             - Return None if not removed.
57         """
58         # TO DO: implement cancel_booking logic
59
60     def move_to_end(self, customer_id):
61         """
62             Move a customer with the given ID to the end of the queue.
63             - Return customer_name if moved.
64             - Return None if queue is empty, customer not found, or already at rear.
65             - Update front/rear if needed.
66         """
67         # TO DO: implement move_to_end logic
68
69     def find_position(self, customer_id):
70         """
71             Find the position of a customer in the queue by ID.
72             - Return (position, customer_name) if found.
73             - Return None if not found.
74         """
75         # TO DO: implement find_position logic
76
77
78 def test():
79     YELLOW = "\033[93m"
80     RESET = "\033[0m"
81
82     queue = TicketQueue()
83
84     # Add initial customers
85     customers = [(101, "Raj"), (102, "Maya"), (103, "Liam"), (104, "Sara")]
86     print(f"\n{YELLOW}Adding customers to the queue:{RESET}")
87     for cust_id, cust_name in customers:
88         added_name = queue.enqueue(cust_name, cust_id)
89         print(f"{added_name} (ID: {cust_id}) joined the queue.")
90
91     print(f"\n{YELLOW}Peek at the next customer:{RESET}")
92     next_customer = queue.peek_next_customer()
93     print(f"Next customer to be served: {next_customer}" if next_customer else "Queue is empty.")
94
95     print(f"\n{YELLOW}Find positions by ID:{RESET}")
96     for cust_id in [102, 104]:
97         result = queue.find_position(cust_id)
98         if result:
99             pos, name = result
100            print(f"{name} (ID {cust_id}) is at position {pos} in the queue.")
101        else:
102            print(f"Customer ID {cust_id} not found.")
103
104    print(f"\n{YELLOW}Move a customer to the end:{RESET}")
105    moved_name = queue.move_to_end(102)
106    print(f"{moved_name} has been moved to the end of the queue." if moved_name else "Customer not found or already at the end.")
107
108    print(f"\n{YELLOW}Cancel a booking:{RESET}")
109    cancelled_name = queue.cancel_booking(103)
110    print(f"{cancelled_name}'s booking has been cancelled." if cancelled_name else "Customer not found.")
111
112    print(f"\n{YELLOW}Serve the next customer:{RESET}")

```

```

113 served_name = queue.dequeue()
114 print(f"{served_name} has been served." if served_name else "Queue is empty.")
115
116 print(f"\n{YELLOW}Peek at the next customer after serving one:{RESET}")
117 next_customer = queue.peek_next_customer()
118 print(f"Next customer to be served: {next_customer}" if next_customer else "Queue
is empty.")
119
120 print(f"\n{YELLOW}Total customers remaining:{RESET} {queue.total_customers()}")
121
122 # Add a new customer
123 print(f"\n{YELLOW}Adding a new customer:{RESET}")
124 added_name = queue.enqueue("Rajat", 105)
125 print(f"{added_name} (ID: 105) joined the queue.")
126
127 # Serve next customer
128 print(f"\n{YELLOW}Serve the next customer:{RESET}")
129 served_name = queue.dequeue()
130 print(f"{served_name} has been served." if served_name else "Queue is empty.")
131
132 # Move last customer to end
133 print(f"\n{YELLOW}Move last customer to end:{RESET}")
134 moved_name = queue.move_to_end(105)
135 print(f"{moved_name} has been moved to the end." if moved_name else "Customer
already at the end or not found.")
136
137 # Peek at next customer
138 print(f"\n{YELLOW}Peek at next customer:{RESET}")
139 next_customer = queue.peek_next_customer()
140 print(f"Next customer to be served: {next_customer}" if next_customer else "Queue
is empty.")
141
142 print(f"\n{YELLOW}Total customers remaining in the queue:{RESET} {queue.
total_customers()}")
143
144 # Run the test function
145 test()

```

Sample Output:

On the next page is a screenshot of the expected program output after implementing all functions correctly:

Adding customers to the queue:
Raj (ID: 101) joined the queue.
Maya (ID: 102) joined the queue.
Liam (ID: 103) joined the queue.
Sara (ID: 104) joined the queue.

Peek at the next customer:
Next customer to be served: Maya

Find positions by ID:
Maya (ID 102) is at position 2 in the queue.
Sara (ID 104) is at position 4 in the queue.

Move a customer to the end:
Maya has been moved to the end of the queue.

Cancel a booking:
Liam's booking has been cancelled.

Serve the next customer:
Raj has been served.

Peek at the next customer after serving one:
Next customer to be served: Maya

Total customers remaining: 2

Adding a new customer:
Rajat (ID: 105) joined the queue.

Serve the next customer:
Sara has been served.

Move last customer to end:
Customer already at the end or not found.

Peek at next customer:
Next customer to be served: Rajat

Total customers remaining in the queue: 2