Leland Wendel

CS 260

Assignment 3 – Linked Queues

Design, implement, and test a Queue data structure that:

1. **Uses a linked-list to store values in the queue**

```
1  // Define a node for the linked queue
2  struct Node {
3      int data;
4      Node* next;
5      Node(int value) : data(value), next(nullptr) {}
6  };
```

(^ from "node.h")

Defines a node struct to represent a node in the linked queue.

```
8  // Define the LinkedQueue class
9  class LinkedQueue {
10
11     private:
12         Node* head; // Points to the head of the queue
13         Node* tail;  // Points to the tail of the queue
14
15     public:
16         LinkedQueue() : head(nullptr), tail(nullptr) {}
```

(^ from "linked_queue.h")

Defines a class 'LinkedQueue' to represent a queue as a linked list.

2. **Has an enqueue method that will appropriately add a value to the back of the queue as an appropriate element**

```
19      void enqueue(int value) {
20          Node* new_node = new Node(value);
21          if (isEmpty()) {
22              head = tail = new_node;
23          } else {
24              tail->next = new_node;
25              tail = new_node;
26          }
27      }
```

(^ from "linked_queue.h")

Enqueue function to add an integer value to the tail of the queue.

3. **Has a dequeue method that will appropriately remove an element from the front of the queue and return its value**

```
30      int dequeue() {
31          if (isEmpty()) {
32              cout << "Queue is empty!" << endl;
33              exit(EXIT_FAILURE);
34          }
35          int value = head->data;
36          Node* temp = head;
37          head = head->next;
38          delete temp;
39          if (head == nullptr) {
40              tail = nullptr;
41          }
42          return value;
43      }
```

(^ from "linked_queue.h")

Dequeue function to remove a value from the head of the queue and return its value.

4. **Optionally has a peek method that returns the value at the front of the queue without removing it**

```
51          int peek() {
52              if (isEmpty()) {
53                  cout << "Queue is empty!" << endl;
54                  exit(EXIT_FAILURE);
55              }
56              return head->data;
57          }
58      };
```

(^ from "linked_queue.h")

Peek function to return the value at the head of the queue without removing it.

5. **Tests: Be sure to include at least one test for each piece of functionality that should verify that your code is working!**

```
12      test.enqueue(1);
13      test.enqueue(2);
14      test.enqueue(3);
```

(from "linked_queue_driver.cpp")

Enqueue integers 1, 2, & 3 into the queue 'test'.

```
16      // Test peek
17      cout << "head element: " << test.peek() << endl;
```

(from "linked_queue_driver.cpp")

Print the value of the head element using the peek function without removing it.

```
19      // Test dequeue
20      cout << "Dequeued elements: ";
21      while (!test.isEmpty()) {
22          cout << test.dequeue() << " ";
23      }
24      cout << endl;
```

(from "linked_queue_driver.cpp")

Dequeue all the values until the list is empty and print the dequeued values.

```
26      // Test isEmpty after dequeue
27      cout << "Is queue empty? " << (test.isEmpty() ? "Yes" : "No") << endl;
28
```

(from "linked_queue_driver.cpp")

Check that the queue is empty after dequeuing all elements and prints the yes or no result.

Output:

```
head element: 1
Dequeued elements: 1 2 3
Is queue empty? Yes
```