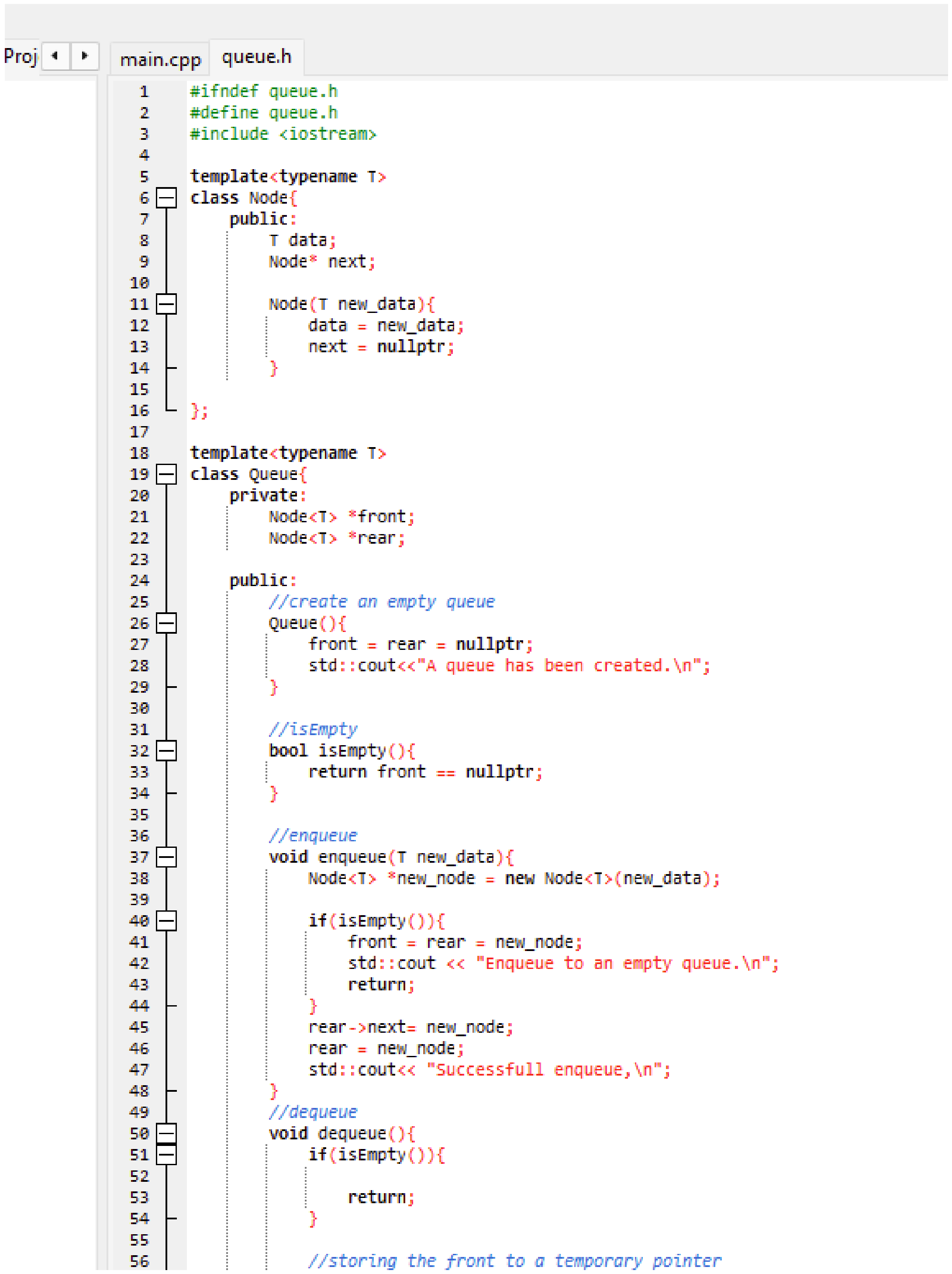
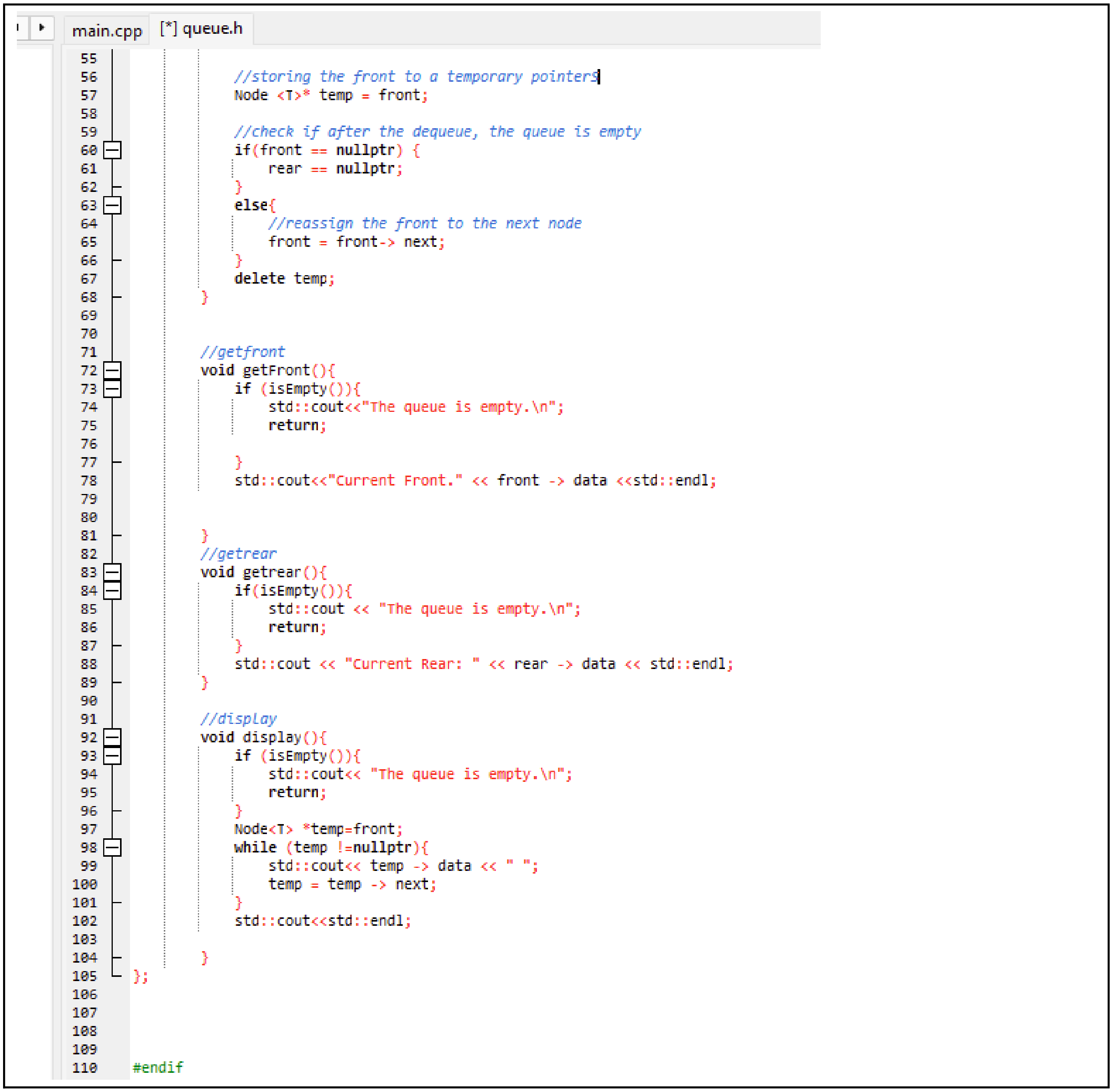
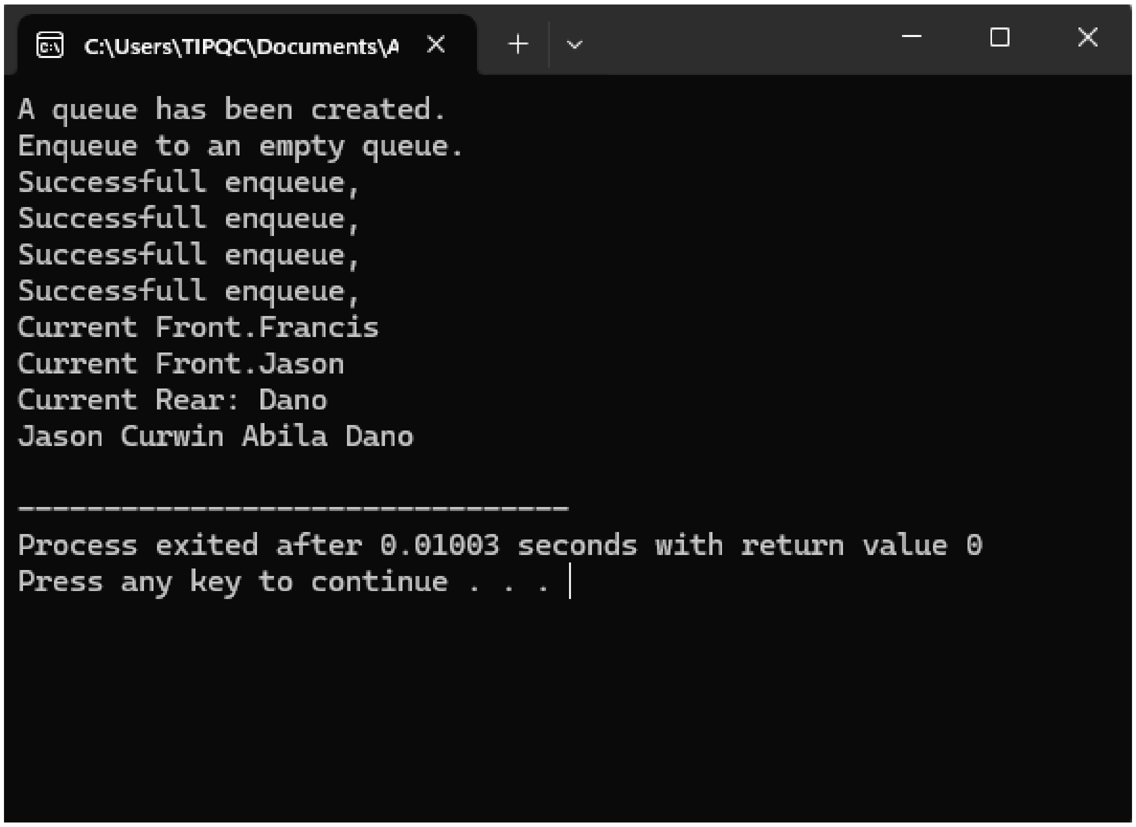
Seatwork 5.1	
Queue - Linked List Application	
Course Code: CPE010	Program: BSCPE
Course Title: Data Structures and Algorithms	Date Performed: 9/9/2025
Section: CPE21S4	Date Submitted: 9/9/2025
Name(s): AVILA, VINCE GABRIEL V.	Instructor: Engr. Jimlord Quejado
6. Output	



```cpp
#ifndef queue.h
#define queue.h
#include <iostream>

template<typename T>
class Node{
    public:
        T data;
        Node* next;

        Node(T new_data){
            data = new_data;
            next = nullptr;
        }

};

template<typename T>
class Queue{
    private:
        Node<T> *front;
        Node<T> *rear;

    public:
        //create an empty queue
        Queue(){
            front = rear = nullptr;
            std::cout<<"A queue has been created.\n";
        }

        //isEmpty
        bool isEmpty(){
            return front == nullptr;
        }

        //enqueue
        void enqueue(T new_data){
            Node<T> *new_node = new Node<T>(new_data);

            if(isEmpty()){
                front = rear = new_node;
                std::cout << "Enqueue to an empty queue.\n";
                return;
            }
            rear->next= new_node;
            rear = new_node;
            std::cout<< "Successfull enqueue,\n";
        }
        //dequeue
        void dequeue(){
            if(isEmpty()){

                return;
            }

            //storing the front to a temporary pointer
```


main.cpp [*] queue.h

```

            //storing the front to a temporary pointers
            Node <T>* temp = front;

            //check if after the dequeue, the queue is empty
            if(front == nullptr) {
                rear == nullptr;
            }
            else{
                //reassign the front to the next node
                front = front-> next;
            }
            delete temp;
        }


        //getfront
        void getFront(){
            if (isEmpty()){
                std::cout<<"The queue is empty.\n";
                return;

            }
            std::cout<<"Current Front." << front -> data <<std::endl;


        }
        //getrear
        void getrear(){
            if(isEmpty()){
                std::cout << "The queue is empty.\n";
                return;
            }
            std::cout << "Current Rear: " << rear -> data << std::endl;
        }

        //display
        void display(){
            if (isEmpty()){
                std::cout<< "The queue is empty.\n";
                return;
            }
            Node<T> *temp=front;
            while (temp !=nullptr){
                std::cout<< temp -> data << " ";
                temp = temp -> next;
            }
            std::cout<<std::endl;

        }
};




#endif
```



```
A queue has been created.
Enqueue to an empty queue.
Successfull enqueue,
Successfull enqueue,
Successfull enqueue,
Successfull enqueue,
Current Front.Francis
Current Front.Jason
Current Rear: Dano
Jason Curwin Abila Dano

--------------------------------
Process exited after 0.01003 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity

Node class - It let us store a pieces of data
Front & Rear - where the dequeue only has a one node that would reset its the front and rear to create nullptr.

8. Conclusion

For my conclusion through this activity, I learned the basics of the Queue Linked List Application. In this discussion our proctor gave us a step by step procedure on how to manipulate each part of the code including the Class queue, enqueue, dequeue, getfront, and getrear. Overall this activity let me gain insights about the topic “Queue”.

9. Assessment Rubric