Activity No. <n>	
<Replace with Title>	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 09/09/25**
Section: CPE 010-CPE21S4	**Date Submitted:09/09/25**
Name(s): KERWIN JAN B. CATUNGAL	**Instructor: Jimlord Quejado**

6. Output

CODES:

main.cpp queueh.h

```
1   #ifndef queueh.h
2   #define queueh.h
3   #include <iostream>
4
5   template<typename T>
6   class Node{
7       public:
8           T data;
9           Node* next;
10          Node(T new_data){
11              data = new_data;
12              next= nullptr;
13      }
14  };
15
```

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

            if(isEmpty()){
                front = rear = new_node;
                std::cout<< "Enqueue to and empty qeueue.\n"
                return;
            }
            rear->next = new_node;
            rear = new_node;
            std::cout<<"Successfully enqueue,\n";
        }
```

```
    // dequeue
    void dequeue(){
        if(isEmpty()){

            return;
        }
        //storing the front to a temporary pointer
        Node <T>* temp = front;

        if (front == nullptr){
            rear == nullptr;
        }
        else{
            //reassign the front to the next code
            front = front -> next;
        }
        delete temp;
    }
```

main.cpp queueh.h

```
66      // getfront
67      void  getFront(){
68          if (isEmpty()){
69              std::cout<<"The queue is empty.\n";
70              return;
71          }
72
73          std::cout<<"Current Front." << front -> data <<std::endl;
74      }
75
76      // getrear
77      void getrear(){
78          if(isEmpty()){
79              std::cout<<"The queue is empty.\n";
80              return;
81          }
82          std::cout<<"Current Rear:" << rear -> data <<std::endl;
83
```

```
        }
        // 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
```

OUTPUT:

```
C:\Users\TIPQC\Documents\C

 A queue has been created.
Enqueue to and empty qeueue.
Successfully enqueue,
Successfully enqueue,
Successfully enqueue,
Successfully enqueue,
Current Front.Francis
Current Front.Dano
Current Rear:Francis
Dano  Abila  Curwin  Francis

--------------------------------
Process exited after 0.01701 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity

So here Node class stores a piece of data type T we also have the front and rear where the dequeue here is has only one node like resets the front and rear to nullptr

8. Conclusion

After doing this activity we first started by creating a main cpp file and a header file and defines a template class node that can store data of any type and each node hold a pieces of data. Overall this activity let me gain some knowledge tho I can say Im really not that good with this but I will try my best and study hard.

9. Assessment Rubric