```
#ifndef QUEUE_T_H
#define QUEUE_T_H
template <typename NodeData>
class Queue
{
private:
    struct Node
    {
        NodeData data;  //data in node
        Node *next;     //pointer to next node

        // Default Constructor
        Node() : next(0) {}
        // Explicit Constructor
        Node(const NodeData &theData, Node *const theNext = 0)
            : data(theData), next(theNext) { }
    };
public:
    Queue() : head(0), tail(0) {}
    bool Empty() const { return head == 0; }
    void Enqueue(const NodeData &elem);
    NodeData Dequeue();
    NodeData Head() { return head->data; }
private:
    Node *tail;     // "end" of queue
    Node *head;
};

template <typename NodeData>
void Queue<NodeData>::Enqueue(const NodeData &elem)
{
    Node* temp = new(nothrow) Node(elem);
    assert(temp != NULL);

    // head == tail if head == NULL, so must also be assigned temp
    if (head == NULL)
        head = temp;
    // add temp after current tail
    else
        tail->next = temp;
    // update tail adress to be new temp node
    tail = temp;
}

template <typename NodeData>
NodeData Queue<NodeData>::Dequeue()
{
    assert (!Empty());
    NodeData poppedData = head->data;
    Node *temp = head;
    head = head->next;
    if (head == NULL)
        tail = NULL;
    delete temp;
    return poppedData;
}

#endif
```