```
//
// CPTN278_A5_Queue_Bettle1.h
//
// Hal Bettle
// 16 January 2009
//
// Queue Class Body File.
//
// Queue implementation is dynamicall linked list based.
// Queue uses an External Node.
// Queue has Error checking.
//
#include "CPTN278_A5_Queue_Bettle1.h"

Queue::Queue()
{
    // initialize the queue
    front = 0;
    back = 0;
    count = 0;
}

bool Queue::Is_Empty(void)
{
    if ( count == 0 )
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool Queue::Is_Full(void)
{
    return false;
}

bool Queue::Enqueue(Node *item)
{
    // Need to account for empty or multi node queue
    if ( ! Queue::Is_Full() )
    {
        Node *element = new Node;
        if ( element == 0 )
        {
            return false;
        }
        element->type = item->type;
        element->fee  = item->fee;
        if ( Queue::Is_Empty() )
        {
            element->flink = back;
            element->blink = front;
            front = element;
            back = element;
        }
        else
        {
            element->flink = back;
            element->blink = back->blink;
            back->blink = element;
            back = element;
        }
        count++;
```

```
        return true;
    }
    else
    {
        return false;
    }
}

bool Queue::Dequeue(void)
{
    Node *element;

    // Need to account for one or multi node queue
    if ( ! Queue::Is_Empty() )
    {
        element = front;
        if ( count == 1 )
        {
            front = 0;
            back = 0;
        }
        else
        {
            front = front->blink;
            front->flink = element->flink;
        }
        count--;
        delete element;
        return true;
    }
    else
    {
        return false;
    }
}

Node *Queue::Front(Node *item)
{
    cout << "Front" << endl;

    if ( Queue::Is_Empty() )
    {
        item = 0;
    }
    else
    {
        item->type = front->type;
        item->fee = front->fee;
    }
    return item;
}

Node *Queue::Back(Node *item)
{
    cout << "Back" << endl;

    if ( Queue::Is_Empty() )
    {
        item = 0;
    }
    else
    {
        item->type = back->type;
        item->fee = back->fee;
    }
    return item;
```

```
}

Queue::~Queue()
{
    while ( ! Queue::Is_Empty() )
    {
        Queue::Dequeue();
    }
}
```