若使用循环链表来表示队列,p是链表中的一个指针.试基于此结构给出队列的插入(enqueue)和删除(dequeue)算法,并给出p为何值时队列空.

解答:

链式队列的类定义

```
template class Queue; //链式队列类的前视定义
template class QueueNode { //链式队列结点类定义
    friend class Queue;
    private:
    Type data; //数据域
    QueueNode *link; //链域
    QueueNode ( Type d = 0, QueueNode *l = NULL ) : data (d), link (l) { } //构造函数
};


template class Queue { //链式队列类定义
public:
      Queue ( ) : p ( NULL ) { } //构造函数
      ~Queue ( ); //析构函数
      void EnQueue ( const Type & item ); //将item加入到队列中
      Type DeQueue ( ); //删除并返回队头元素
      Type GetFront ( ); //查看队头元素的值
      void MakeEmpty ( ); //置空队列,实现与~Queue ( ) 相同
      int IsEmpty ( ) const { return p == NULL; } //判队列空否
private:
    QueueNode *p; //队尾指针(在循环链表中)
};

template Queue::~Queue ( ) { //队列的析构函数
      QueueNode *s;
      while ( p != NULL ) { s = p; p = p→link; delete s; } //逐个删除队列中的结点
}

//队列的插入函数
template void Queue::EnQueue ( const Type & item ) {
    if ( p == NULL ) { //队列空, 新结点成为第一个结点
        p = new QueueNode ( item, NULL ); p→link = p;
}
    else { //队列不空, 新结点链入p之后
      QueueNode *s = new QueueNode ( item, NULL );
      s→link = p→link; p = p→link = s; //结点p指向新的队尾
```

```
    }
}

//队列的删除函数
template Type Queue::DeQueue ( ) {
    if ( p == NULL ) { cout << "队列空, 不能删除!" << endl; return 0; }
    QueueNode *s = p; //队头结点为p后一个结点
    p→link = s→link; //重新链接, 将结点s从链中摘下
    Type retvalue = s→data; delete s; //保存原队头结点中的值, 释放原队头结点
    return retvalue; //返回数据存放地址
}

//队空条件 p == NULL.
```