的最大限度有关)。这个问题在商业上很重要，因为研究表明，人们会很快挂上电话。

如果我们有 k 个接线员，那么这个问题解决起来要困难得多。解析地求解起来困难的问题往往使用模拟的方法进行。此时，我们需要使用一个队列来进行模拟。如果 k 很大，那么我们还需要其他一些数据结构来使得模拟更有效地进行。在第 6 章将会看到模拟是如何进行的，那时我们将对 k 的若干值进行模拟并选择能够给出合理等待时间的最小的 k。

正如栈一样，队列还有其他丰富的用途，这样一种简单的数据结构竟然能够如此重要，实在令人惊奇。

小结

本章描述了一些 ADT 的概念，并且利用三种最常见的抽象数据类型(ADT)阐述了这种概念。主要目的就是将抽象数据类型的具体实现与它们的功能分开。程序必须知道操作都做些什么，但是如果不知道如何去做那就更好。

表、栈和队列或许在全部计算机科学中是三个基本的数据结构，大量的例子证明了它们广泛的用途。特别地，我们看到栈是如何用来记录过程和方法调用的，以及递归实际上是如何实现的。这对于我们的理解非常重要，其原因不只因为它使得过程语言成为可能，而且还因为知道递归的实现从而消除了围绕其使用的大量迷团。虽然递归非常强大，但是它并不是完全随意的操作；递归的误用和乱用可能导致程序崩溃。

练习

3.1 给定一个表 L 和另一个表 P，它们包含以升序排列的整数。操作 printLots(L,P)将打印 L 中那些由 P 所指定的位置上的元素。例如，如果 P=1,3,4,6，那么，L 中位于第 1、第 3、第 4 和第 6 个位置上的元素被打印出来。写出过程 printLots(L,P)。只可使用 public 型的 Collections API 容器操作。该过程的运行时间是多少?

3.2 通过只调整链(而不是数据)来交换两个相邻的元素，使用

a. 单链表。

b. 双链表。

3.3 实现 MyLinkedList 的 contains 例程。

3.4 给定两个已排序的表 L_1 和 L_2，只使用基本的表操作编写计算 $L_1 \cap L_2$ 的过程。

3.5 给定两个已排序的表 L_1 和 L_2，只使用基本的表操作编写计算 $L_1 \cup L_2$ 的过程。

3.6 Josephus 问题(Josephus problem)是下面的游戏：N 个人编号从 1 到 N，围坐成一个圆圈。从 1 号开始传递一个热土豆。经过 M 次传递后拿着热土豆的人被清除离座，围坐的圆圈缩紧，由坐在被清除的人后面的人拿起热土豆继续进行游戏。最后剩下的人取胜。因此，如果 $M=0$ 和 $N=5$，则游戏人依序被清除，5 号游戏人获胜。如果 $M=1$ 和 $N=5$，那么被清除的人的顺序是 2,4,1,5。

a. 编写一个程序解决 M 和 N 在一般值下的 Josephus 问题，应使程序尽可能地高效率，要确保能够清除各个单元。

b. 你的程序的运行时间是多少?

3.7 下列程序的运行时间是多少?

```
public static List<Integer> makeList( int N )
{
```