V. Queues

[Most of the details about queues are left for to read about and work out in Lab 6.]

Def. As a data structure, a **queue** is an ordered collection of data items with the property that items can be removed only at one end, called the **front** of the queue, and items can be added only at the other end, called the **back** of the queue. Basic operations are:

- **construct**: Create an empty queue
- **empty**: Check if a queue is empty
- **addQ**: Add a value at the back of the queue
- **front**: Retrieve the value at the front of the queue
- **removeQ**: Remove the value at the front of the queue

Whereas a stack is a Last-In-First-Out (LIFO) structure, a queue is a **First-In-First-Out (FIFO)** or **First-Come-First-Served (FCFS)** structure.

2. Examples:
   a. I/O buffers: queues, scrolls, deques

   → From a file: *(queue)*

   ![Diagram of queue operation from file]

   → Interactively: *(scroll — queue on one end, stack on the other)*

   ![Diagram of queue operation interactively]
Screen handling: (*deque* — double-ended queue)

b. Scheduling queues in a multi-user computer system:

→ Printer queue: When files are submitted to a printer, they are placed in the printer queue. The printer software executes an algorithm something like:

```java
for (;;)
{
    while (printerQueue.empty())
        sleep 1;
    printFile = printerQueue.removeQ();
    Print(printFile);
}
```

→ Other Queues:
  Resident queue: On disk, waiting for memory
  Ready queue: In memory — has everything it needs to run, except the CPU
  Suspended queue: Waiting for I/O transfer or to be reassigned the CPU
c. CPU Scheduling: Probably uses a priority queue: Items with lower priority are behind all those with higher priority. 

(Usually a new item is inserted behind those with the same priority.)