(a) First Part

```c
if (token_present)
  clock++; // Prelude
  broadcast (Request, clock, i);
  wait (access, token);
  token_present = true;

token_hold = true;
<critical section>:
  token[i] = clock;
  token_hold = false;
for (int i = 1; i < n; i++)
  if (request(j) > token[j] && token_present)
    token_present = false;
    send (access, token[j]);
for (j = 1; j < t-1; j++)
  if (request(i) > token[i] && token_present)
    token_present = false;
    send(access, token[i]);
```

(b) Second Part

```c
if (received (Request, k, i)
  request (i) = max (request(i), k);
if (token_present && token_hold)
  <text of postlude>
```

**Notation**
- `send (j, access, token)` end message of type access, with token, by process j
- `broadcast (request, clock, i)` send message from process i of type request, with time-stamp clock, to all other processes
- `received (request, t, j)` receive message from process j of type request, with time-stamp t

**Figure 15.11** Token-Passing Algorithm (for process Pj)