```
boolean flag [2];
int turn;
void P0()
{
     while (true)
     {
          flag [0] = true;
          while (flag [1])
               if (turn == 1)
          {
               flag [0] = false;
               while (turn == 1)
                    /* do nothing */;
               flag [0] = true;
          }
          /* critical section  */;
          turn = 1;
          flag [0] = false;
          /* remainder   */;
     }
}
void P1( )
{
     while (true)
     {
          flag [1] = true;
          while (flag [0])
          if (turn == 0)
          {
               flag [1] = false;
               while (turn == 0)
                    /* do nothing */;
               flag [1] = true;
          }
          /* critical section   */;
          turn = 0;
          flag [1] = false;
          /* remainder   */;
     }
}
void main ()
{
     flag [0] = false;
     flag [1] = false;
     turn = 1;
     parbegin (P0, P1);
}
```

Figure A.2 Dekker's Algorithm

```
boolean flag [2];
int turn;
void P0()
{
     while (true)
     {
          flag [0] = true;
          turn = 1;
          while (flag [1] && turn == 1)
               /* do nothing */;
          /* critical section  */;
          flag [0] = false;
          /* remainder  */;
     }
}
void P1()
{
     while (true)
     {
          flag [1] = true;
          turn = 0;
          while (flag [0] && turn == 0)
               /* do nothing */;
          /* critical section  */;
          flag [1] = false;
          /* remainder  */
     }
}
void main()
{
     flag [0] = false;
     flag [1] = false;
     parbegin (P0, P1);
}
```

Figure A.3 Peterson's Algorithm for Two Processes