```
program PCON4;

(* producer-consumer problem  - monitor solution *)
monitor BUFFER;

export
     PUT, TAKE;

const
     BUFFMAX = 4;

var
     STORE: array[0..BUFFMAX] of char;
     COUNT: integer;
     NOTFULL, NOTEMPTY: condition;
     NEXTIN, NEXTOUT: integer;
procedure PUT(CH: char);

begin
     if COUNT > BUFFMAX then
          delay(NOTFULL);
     STORE[NEXTIN] := CH;
     COUNT := COUNT + 1;
     NEXTIN := (NEXTIN + 1) mod (BUFFMAX + 1);
     resume(NOTEMPTY)
end;  (* PUT *)
procedure TAKE(var CH: char);

begin
     if COUNT = 0 then
          delay(NOTEMPTY);
     CH := STORE[NEXTOUT];
     COUNT := COUNT - 1;
     NEXTOUT := (NEXTOUT + 1) mod (BUFFMAX + 1);
     resume(NOTFULL)
end;  (* TAKE *)
begin  (* body of BUFFER *)
     COUNT := 0;
     NEXTIN := 0;
     NEXTOUT := 0
end;  (* BUFFER *)
```

```
process PRODUCER;

var
     LOCAL: char;

begin
     for LOCAL := 'a' to 'z' do
          BUFFER.PUT(LOCAL);
end;  (* PRODUCER *)
```

```
process CONSUMER;

var
     CH: char;

begin
     repeat
          BUFFER.TAKE(CH);
          write(CH)
     until CH = 'z';
     writeln
end;  (* CONSUMER *)
```

```
begin  (* main *)
     cobegin
          PRODUCER;
          CONSUMER
     coend
end.
```

5.5. Process States and Monitors

This section summarises the effects on process state of the features described in this chapter.

1. A process that attempts to enter a monitor that is already occupied becomes "suspended" on the monitor boundary queue.
2. A process that executes a `delay` becomes "suspended" on the named condition.
3. A process that executes a `resume` *that has the effect of unsuspending a process* becomes "suspended" on the monitor chivalry queue.
4. A process suspended on a condition can be made "executable" by the `resume` operation.
5. A process suspended on a monitor boundary queue can be made executable by a process leaving the monitor.