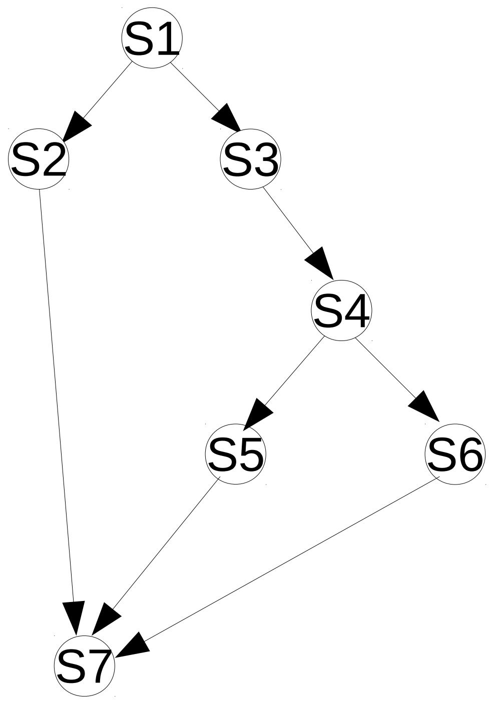
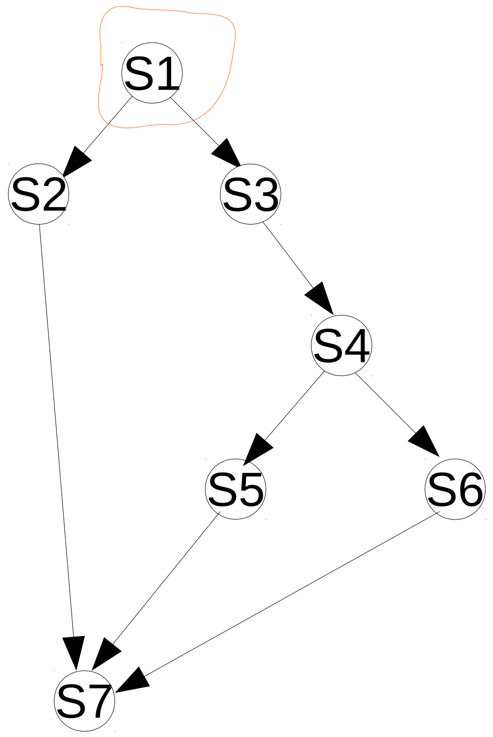
Precedence Graph and Implementation

CS303
6 Sep 2018

Precedence Graph with Single Confluence Point in Concurrency

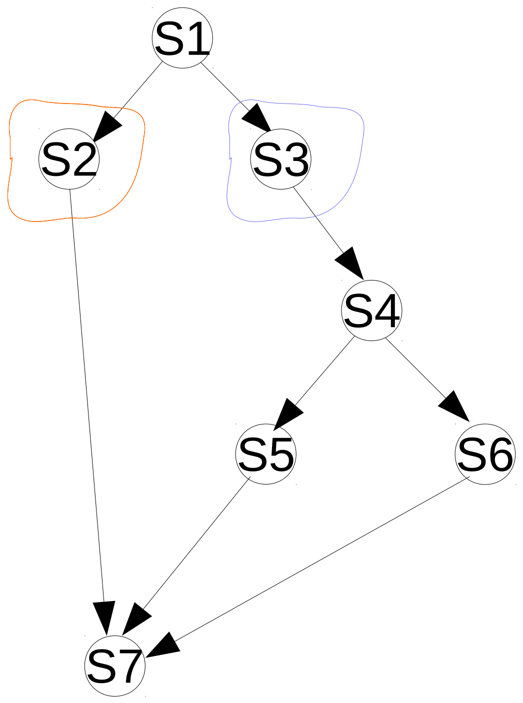


Precedence Graph: Implementing in the Code with fork-join constructs



```
Begin
    S1
```

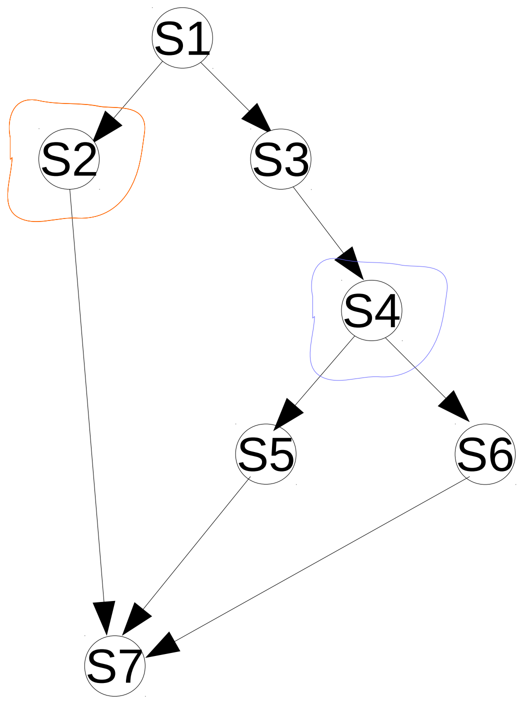
Precedence Graph: Implementing in the Code with fork-join constructs



```
Begin
    S1
    fork L1
    S2
```

```
L1: S3
```

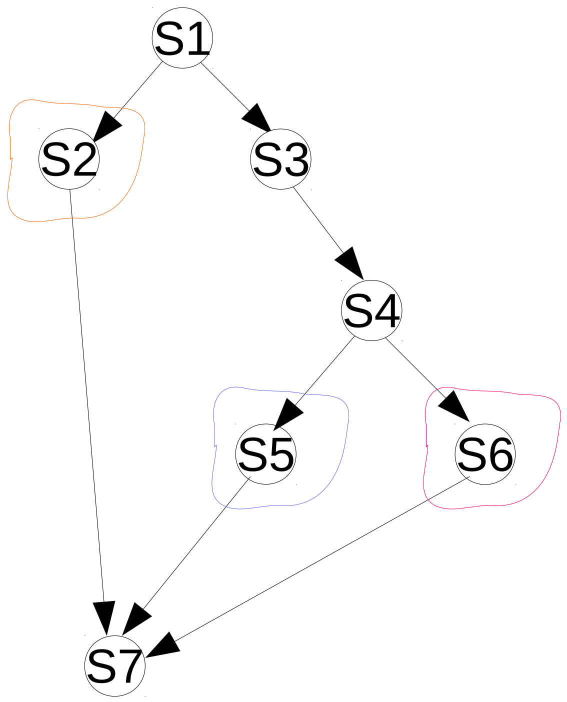
Precedence Graph: Implementing in the Code with fork-join constructs



```
Begin
    S1
    fork L1
    S2
```

```
L1: S3
    S4
```

Precedence Graph: Implementing in the Code with fork-join constructs

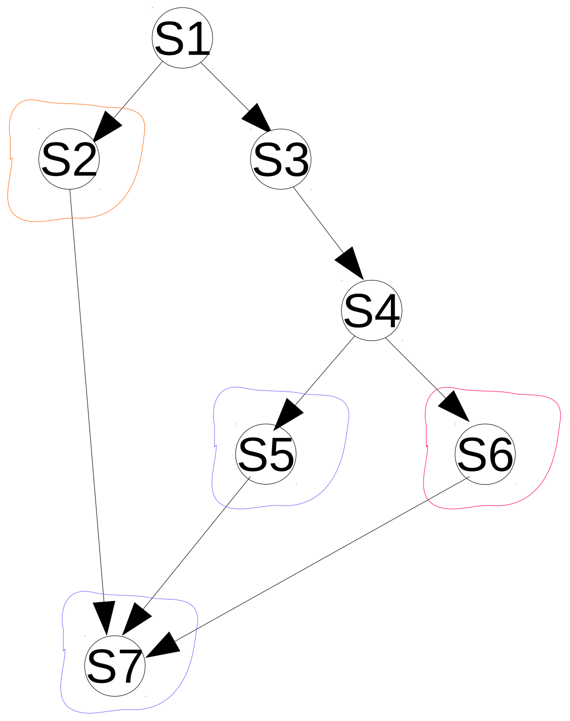


```
Begin
    S1
    fork L1
    S2
```

```
L1: S3
    S4
    fork L2
```

```
L2: S6
```

Precedence Graph: Implementing in the Code with fork-join constructs

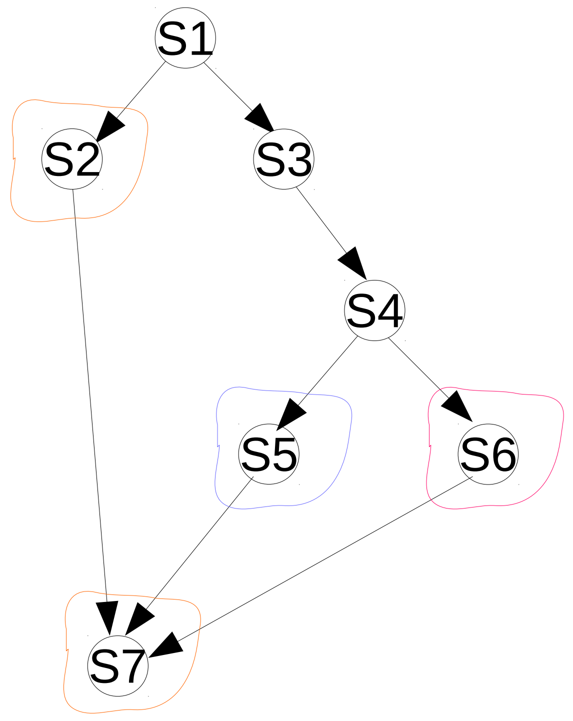


```
Begin
    S1
    fork L1
    S2
    join
    S7
End
```

```
L1: S3
    S4
    fork L2
```

```
L2: S6
```

Precedence Graph: Implementing in the Code with fork-join constructs

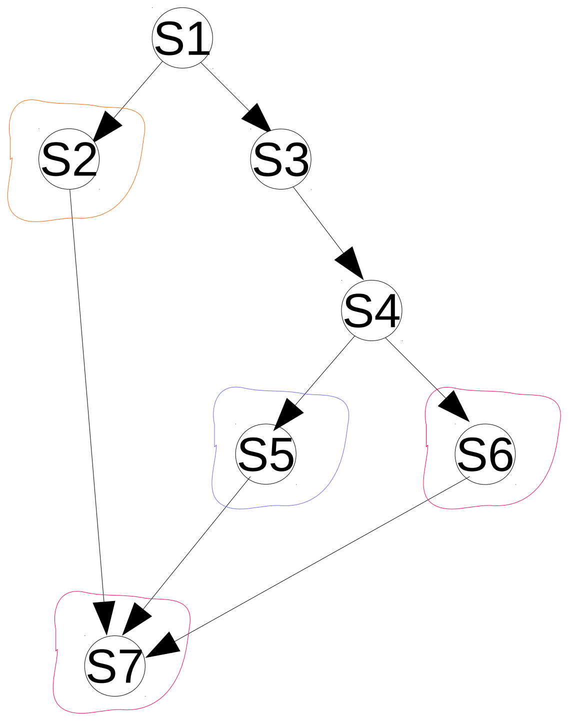


```
Begin
    S1
    fork L1
    S2
    join
    S7
End
```

```
L1: S3
    S4
    fork L2
```

```
L2: S6
```

Precedence Graph: Implementing in the Code with fork-join constructs

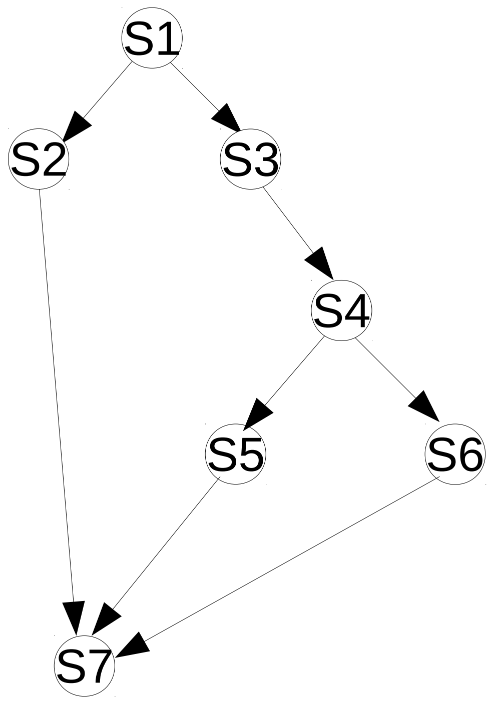


```
Begin
    S1
    fork L1
    S2
    L3: join
    S7
End
```

```
L1: S3
    S4
    fork L2
    S5
    goto L3
```

```
L2: S6
goto L3
```

Precedence Graph: Implementing in the Code with fork-join constructs

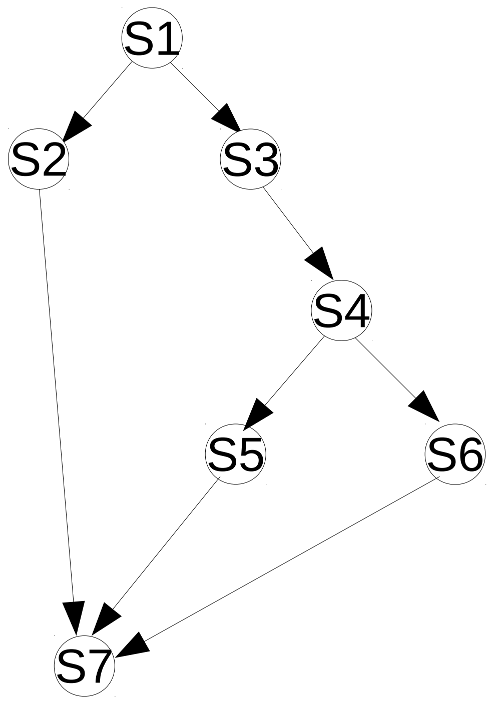


```
Begin
    count = 3
    S1
    fork L1
    S2
    L3: join(count)
    S7
End
```

```
L1: S3
    S4
    fork L2
    S5
    goto L3
```

```
L2: S6
goto L3
```

Precedence Graph: Implementing in the Code with fork-join constructs

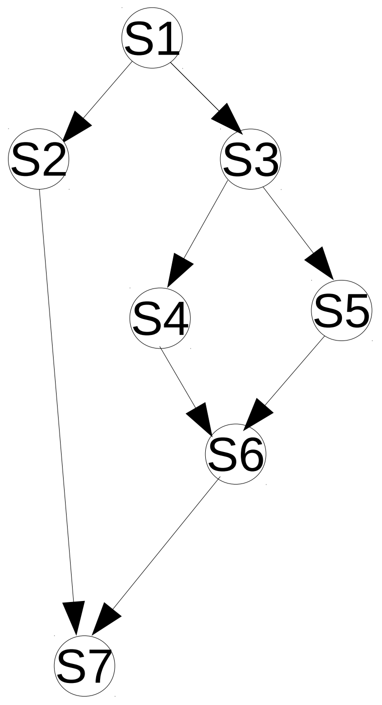


```
Begin
    count = 3
    S1
    fork L1
    S2
    L3: join(count)
    S7
End

join(count)
    {
    count--
    if count>0 then QUIT
    }
```

```
L1: S3
    S4
    fork L2
    S5
    goto L3
```

```
L2: S6
goto L3
```

Precedence Graph with MULTIPLE JOINS in Concurrency

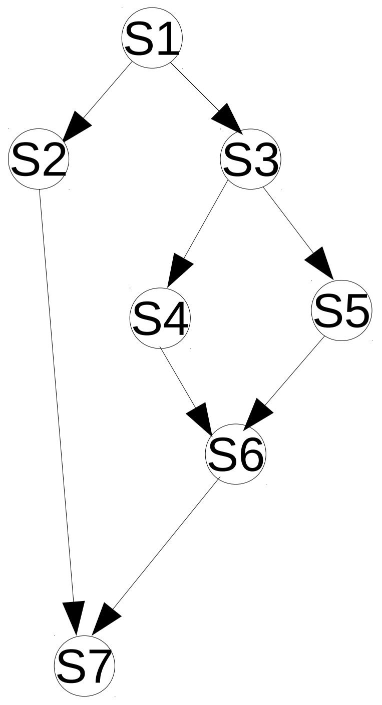


```
Begin
    count1 = 2
    Count2 = 2
    S1
    fork L1
    S3
    fork L2
    S5
    L3: join(count1)
    S6
    L4: join(count2)
    S7
End
```

```
L1: S2
    goto L4
```

```
L2: S4
goto L3
```

Precedence Graph with MULTIPLE JOINS in Concurrency

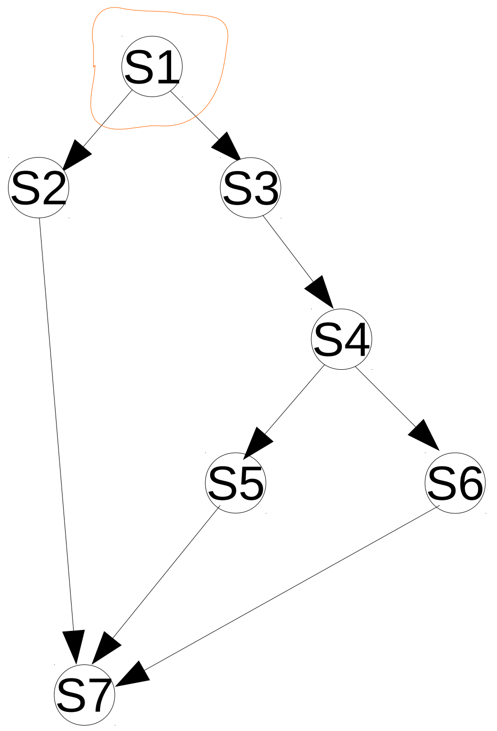


```
Begin
    count1 = 2
    Count2 = 2
    S1
    fork L1
    S3
    fork L2
    S5
    L3: join(count1)
    S6
    L4: join(count2)
    S7
End
```

```
L1: S2
    goto L4
```

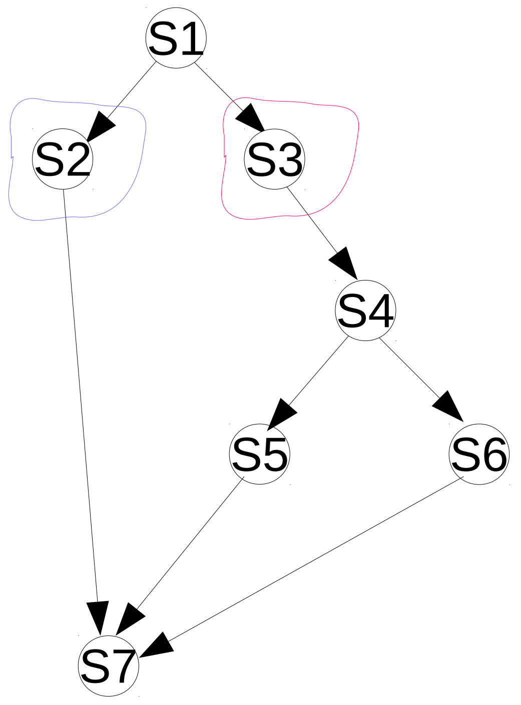
```
L2: S4
goto L3
```

Precedence Graph: Implementing in the Code with **cobegin-coend**



```
Begin
    S1;



End
```

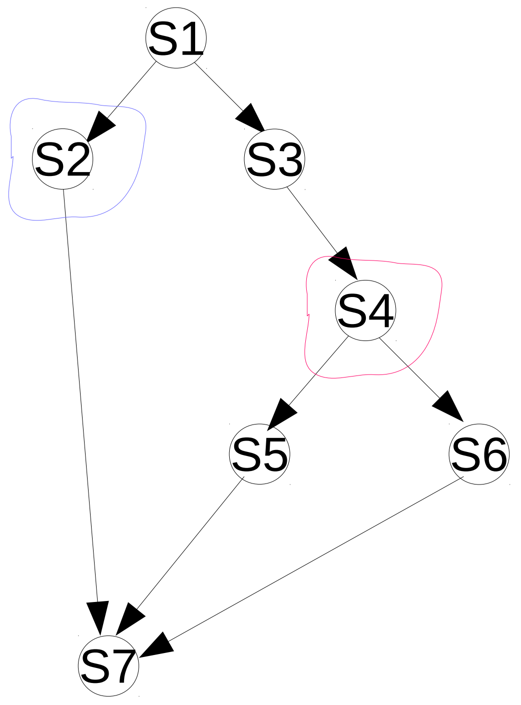
Precedence Graph: Implementing in the Code with **cobegin-coend**



```
Begin
    S1;
    cobegin
        S2;
        begin
        S3;





End
```

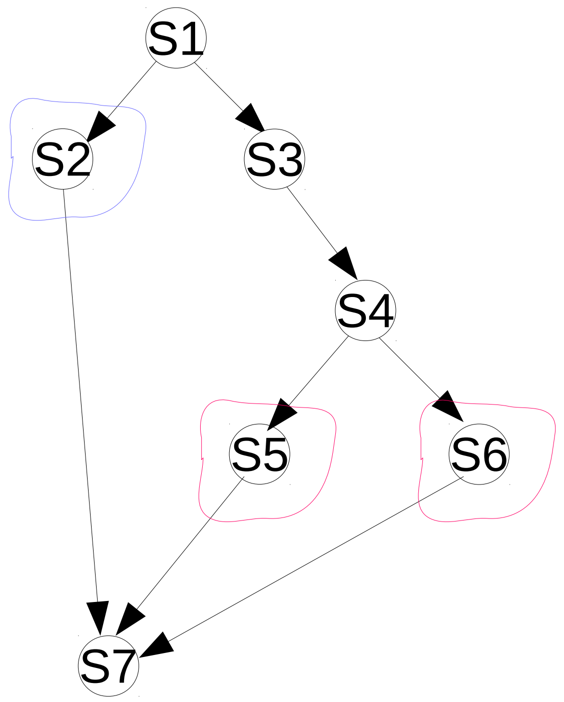
(In the code above, **cobegin** and **begin** are bold.)

Precedence Graph: Implementing in the Code with **cobegin-coend**



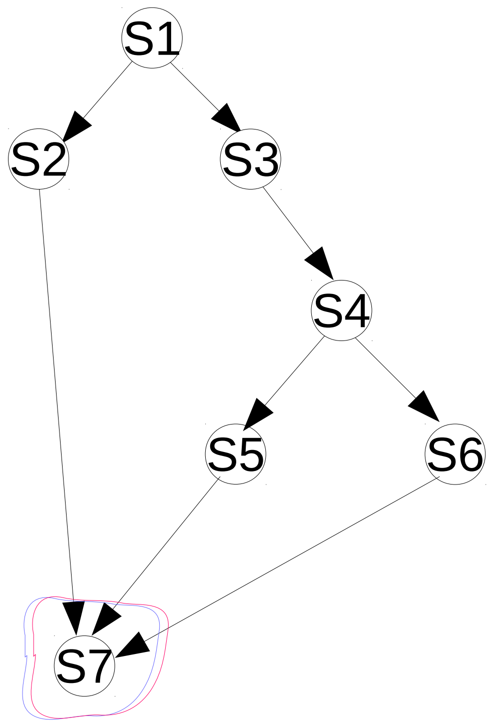
```
Begin
    S1;
    cobegin
        S2;
        begin
            S3;
            S4;



End
```

Precedence Graph: Implementing in the Code with **cobegin-coend**



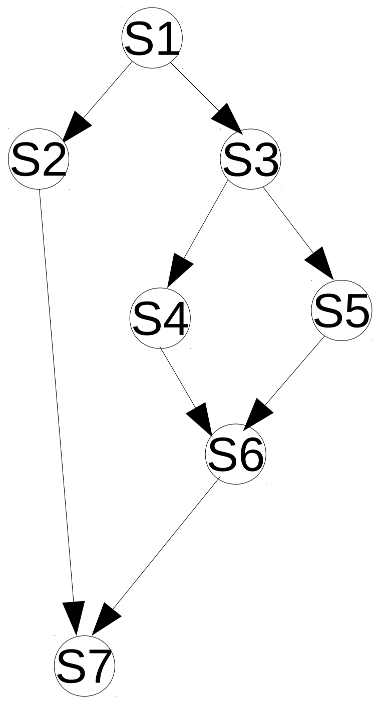
```
Begin
    S1;
    cobegin
        S2;
        begin
            S3;
            S4;
            cobegin
                S5;
                S6;
            coend
        end

End
```

Precedence Graph: Implementing in the Code with **cobegin-coend**



```
Begin
    S1;
    cobegin
        S2;
        begin
            S3;
            S4;
            cobegin
                S5;
                S6;
            coend
        end
    S7;
End
```

Precedence Graph with MULTIPLE JOINS in Concurrency



```
Begin
    S1
        cobegin
            S2
            begin
                S3
                 cobegin
                     S4
                     S5
                 coend
                S6
            end
        coend
    S7
End
```