◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```

◇ Peterson's Algorithm ◇

```
program peterson;
(* Peterson's two-process mutual exclusion algorithm *)

var
  count, turn : integer;
  flag1, flag2: boolean;

process turnstile1;
var loop: integer;
begin
 for loop := 1 to 20 do
  begin
   (* entry protocol *)
   flag1:= true;    (* announce intent to enter *)
   turn:= 2;      (* give priority to other process *)
   while flag2 and (turn = 2) do
    null;
   (* end of entry protocol *)
   (* critical section *)
   count := count + 1;
   (* end of critical section *)
   (* exit protocol *)
   flag1:= false
   (* end of exit protocol *)
  end
end;

process turnstile2; (* similar *)
```