MUTUAL EXCLUSION ALGORITHMS FOR TWO PROCESSES

```
int turn;
main()
...
$turn \leftarrow 1$;
$threadCreate(funcOne)$;
$threadCreate(funcTwo)$;
...
return;

funcOne()
while $TRUE$ do
    while $turn == 2$ do
    end

    /* Critical Section                                      */
     $turn \leftarrow 2$;
    /* Remainder Section                                     */

end
return;

funcTwo()
while $TRUE$ do
    while $turn == 1$ do
    end

    /* Critical Section                                      */
     $turn \leftarrow 1$;
    /* Remainder Section                                     */

end
return;
```

Algorithm 1: 1st Attempt - Mutual Exclusion

```
/* flag[i] indicates whether func i is inside its critical section */
boolean flag[2];
main()
...
$flag[1] \leftarrow FALSE$;
$flag[2] \leftarrow FALSE$;
$threadCreate(funcOne)$;
$threadCreate(funcTwo)$;
...
return;

funcOne()
while $TRUE$ do
    while $flag[2]$ do

    end
    $flag[1] \leftarrow TRUE$;
    /* Critical Section                                            */
    $flag[1] \leftarrow FALSE$;
    /* Remainder Section                                           */

end
return;

funcTwo()
while $TRUE$ do
    while $flag[1]$ do

    end
    $flag[2] \leftarrow TRUE$;
    /* Critical Section                                            */
    $flag[2] \leftarrow FALSE$;
    /* Remainder Section                                           */

end
return;
```

Algorithm 2: 2nd Attempt - Mutual Exclusion

```
/* flag[i] indicates whether func i is ready to enter its critical
   section                                                        */
boolean flag[2];
main()
...
$flag[1] \leftarrow FALSE$;
$flag[2] \leftarrow FALSE$;
$threadCreate(funcOne)$;
$threadCreate(funcTwo)$;
...
return;

funcOne()
while $TRUE$ do
    $flag[1] \leftarrow TRUE$;
    while $flag[2]$ do
    end
    /* Critical Section                                           */
    $flag[1] \leftarrow FALSE$;
    /* Remainder Section                                          */
end
return;

funcTwo()
while $TRUE$ do
    $flag[2] \leftarrow TRUE$;
    while $flag[1]$ do
    end
    /* Critical Section                                           */
    $flag[2] \leftarrow FALSE$;
    /* Remainder Section                                          */
end
return;
```

Algorithm 3: 3rd Attempt - Mutual Exclusion

```
/* flag[i] indicates if func i is ready to enter critical section   */
boolean flag[2];
main()
...
$flag[1] \leftarrow FALSE$;
$flag[2] \leftarrow FALSE$;
$threadCreate(funcOne)$;
$threadCreate(funcTwo)$;
...
return;

funcOne()
while $TRUE$ do
    $flag[1] \leftarrow TRUE$;
    while $flag[2]$ do
        $flag[1] \leftarrow FALSE$;
        $sleep(RandomTimeInterval)$;
        $flag[1] \leftarrow TRUE$;
    end
    /* Critical Section                                              */
    $flag[1] \leftarrow FALSE$;
    /* Remainder Section                                             */
end
return;

funcTwo()
while $TRUE$ do
    $flag[2] \leftarrow TRUE$;
    while $flag[1]$ do
        $flag[2] \leftarrow FALSE$;
        $sleep(RandomTimeInterval)$;
        $flag[2] \leftarrow TRUE$;
    end
    /* Critical Section                                              */
    $flag[2] \leftarrow FALSE$;
    /* Remainder Section                                             */
end
return;
```

Algorithm 4: 4th Attempt - Mutual Exclusion

```
/* flag[i] indicates whether func i is ready to enter its critical
   section                                                          */
int turn;
boolean flag[2];
main()
...
$flag[1] \leftarrow FALSE$;
$flag[2] \leftarrow FALSE$;
$threadCreate(PetersonOne)$;
$threadCreate(PetersonTwo)$;
...
return;

PetersonOne()
while $TRUE$ do
    $flag[1] \leftarrow TRUE$;
    $turn \leftarrow 2$;
    while $flag[2]$ and $turn == 2$ do
    end
    /* Critical Section                                             */
    $flag[1] \leftarrow FALSE$;
    /* Remainder Section                                            */
end
return;

PetersonTwo()
while $TRUE$ do
    $flag[2] \leftarrow TRUE$;
    $turn \leftarrow 1$;
    while $flag[1]$ and $turn == 1$ do
    end
    /* Critical Section                                             */
    $flag[2] \leftarrow FALSE$;
    /* Remainder Section                                            */
end
return;
```

Algorithm 5: Peterson's Solution - Mutual Exclusion