

Exercise: Synchronisation mechanisms, threads and condition variables in EiffelThread

1. Mutual exclusion with busy-waits

Consider two processes (P1 and P2) that have mutual critical sections. In order to protect access to these critical sections, it can be assumed that each process executes an entry protocol before the critical section and an exit protocol afterwards.

```
process P;  
  loop  
    entry protocol  
    critical section  
    exit protocol  
    non-critical section  
  end  
end P;
```

Give a correct algorithm for the mutual exclusion synchronisation between two processes P1 and P2 using busy-waits and show informally that the algorithm is correct.

2. Multi-threaded application with EiffelThread

Using the EiffelThread library of EiffelStudio 5.7 write a simple concurrent application “racer” which launches several racers, performing n iterations (number of racers and iterations are defined by the user). For instance, you can try to launch 10 racers, running during 5 iterations, resulting in the following output:

```
** Thread race  
** -----  
** # of racers: 10  
** length of race: 5  
** Race started!  
1|--1  
2|---1  
3|----1  
4|-----1  
5|-----1  
1|--2  
2|---2  
3|----2  
4|-----2  
5|-----2
```

```
1|--3
2|---3
3|----3
4|-----3
5|-----3
1|--4
2|---4
3|----4
4|-----4
5|-----4
1|--5
2|---5
3|----5
4|-----5
5|-----5
1|--6
2|---6
3|----6
1|--7
2|---7
3|----7
4|-----6
4|-----7
5|-----6
1|--8
2|---8
3|----8
4|-----8
5|-----8
5|-----7
1|--9
1|--10
```

** All runners started

```
2|---10
3|----10
2|---9
3|----9
4|-----9
4|-----10
5|-----10
5|-----9
```

3. Producer-Consumer example in EiffelThread

Implement the Producer-Consumer application using the EiffelThread library of EiffelStudio 5.7. Use condition variables to synchronize consumers and producers to access the shared buffer.