Конкурентно и дистрибуирано програмирање *01. а̄прил 2023.*

1. (12,5) Написати и објасните *Test and Set* алгоритам за критичну секцију (*coarse grain*). Реализовати (*fine grain*) верзију алгоритма уколико би на датом процесору уместо *TS* постојала операција *Compare-And-Swap* која би недељиво обављала (CAS(a, b, c):
< if (a == c) { c = b; return true;}
else { a = c; return false;}>).

2. (12,5) Посматра се сто у мензи за којим може да седи највише N особа (N > 2). Особа узима храну (*getFood()*), седа за сто (*sitAtTable()*), једе (*eat()*), устаје са стола (*leaveTable()*) и одлази (*walkAway()*). Потребно је обезбедити да ниједна особа не једе, тј. седи за столом сама. Такође, свака особа која је узела храну мора да седне за сто и једе. Користећи семафоре написати методе *sitAtTable()* и *leaveTable()* које решавају овај проблем.

*Колоквијум т̄раје **1,5** сат̄.*

Конкурентно и дистрибуирано програмирање *01. а̄прил 2023.*

1. (12,5) Написати и објасните *Test and Set* алгоритам за критичну секцију (*coarse grain*). Реализовати (*fine grain*) верзију алгоритма уколико би на датом процесору уместо *TS* постојала операција *Compare-And-Swap* која би недељиво обављала (CAS(a, b, c):
< if (a == c) { c = b; return true;}
else { a = c; return false;}>).

2. (12,5) Посматра се сто у мензи за којим може да седи највише N особа (N > 2). Особа узима храну (*getFood()*), седа за сто (*sitAtTable()*), једе (*eat()*), устаје са стола (*leaveTable()*) и одлази (*walkAway()*). Потребно је обезбедити да ниједна особа не једе, тј. седи за столом сама. Такође, свака особа која је узела храну мора да седне за сто и једе. Користећи семафоре написати методе *sitAtTable()* и *leaveTable()* које решавају овај проблем.

*Колоквијум т̄раје **1,5** сат̄.*