Κωδικός Εργασίας (3.1)

**Ομάδα** [2]

**Μέλος A**: [Κατερίνα, Καράκουλα, 1604, karakoul@uth.gr]

**Μέλος B**: [Θεοδοσίου, Κωνσταντίνος, 1619, konstheo@uth.gr]

## Περιγραφή σημείων συγχρονισμού με ψευτοκώδικα

|  |
| --- |
| (α) Για τα mtx\_blue, cond\_blue. Η λειτουργια που θελουμε να επιτευχθει ειναι ανα Ν αυτοκινητα να αλλαζει η κατευθυνση στη γεφυρα και τα αυτοκινητα να επικοινωνουν μεταξυ τους για το ποτε εχουν την αδεια να ανεβουν στη γεφυρα. Η υλοποιηση μας βασιζεται στο ξεκλειδωμα Ν αμαξιων που θα παιρνουν την αδεια για να ανεβουν στη γεφυρα απο το πρωτο αμαξι της ομαδας τους, το οποιο ενεργοποιειται απο το τελευταιο αμαξι της προηγουμενης ομαδας Ν αμαξιων (το πρωτο απ ολα τα αμαξια που εχει εισαγει ο χρηστης ξεκλειδωνεται οταν ολα τα υπολοιπα αμαξια της κατηγοριας του εχουν μπλοκαριστει) |
| (β) main:  init (mtx\_blue)  init (cond\_blue)  metritis = 0  count\_b = 0  blue thread:  lock (mtx\_blue)  metritis++:  if (metritis == ola\_ta\_blue\_cars) {  signal (cond\_blue)  }  if (ola\_ta\_blue\_cars != 1) {  wait (cond\_blue, mtx\_blue)  }  count\_b++  if ((count\_b-1) % N == 0) {  for (N cars) {  signal (cond\_blue)  }  }  unlock (mtx\_blue)  if (no\_reds && count\_b%N == 0 && count\_b!= ola\_ta\_blue\_cars) {  signal(cond\_blue)  }  red thread :  if(count\_r%N == 0 && ola\_ta\_blue\_cars != count\_b) {  signal(blue)  } |

## Περιγραφή σημείων συγχρονισμού με ψευτοκώδικα

|  |
| --- |
| (α) Για τα mtx\_blue, cond\_blue. Η λειτουργια που θελουμε να επιτευχθει ειναι ανα Ν αυτοκινητα να αλλαζει η κατευθυνση στη γεφυρα και τα αυτοκινητα να επικοινωνουν μεταξυ τους για το ποτε εχουν την αδεια να ανεβουν στη γεφυρα. Η υλοποιηση μας βασιζεται στο ξεκλειδωμα Ν αμαξιων που θα παιρνουν την αδεια για να ανεβουν στη γεφυρα απο το πρωτο αμαξι της ομαδας τους, το οποιο ενεργοποιειται απο το τελευταιο αμαξι της προηγουμενης ομαδας Ν αμαξιων (το πρωτο απ ολα τα αμαξια που εχει εισαγει ο χρηστης ξεκλειδωνεται οταν ολα τα υπολοιπα αμαξια της κατηγοριας του εχουν μπλοκαριστει). |
| (β) main:  init (mtx\_red)  init (cond\_red)  metritis = 0  count\_r = 0  red thread:  lock (mtx\_red)  if(i\_blue == 0) {  metritis++:  if (metritis == ola\_ta\_red\_cars) {  signal (cond\_red)  }  if (ola\_ta\_red\_cars != 1) {  wait (cond\_red, mtx\_red)  }  }  else {  wait (cond\_red, mtx\_red)  }  count\_r++  if ((count\_r-1) % N == 0) {  for (N cars) {  signal (cond\_red)  }  }  unlock (mtx\_red)  if (no\_blues && count\_r%N == 0 && count\_r!= ola\_ta\_red\_cars) {  signal(cond\_red)  }  blue thread :  if(count\_b%N == 0 && ola\_ta\_red\_cars != count\_r) {  signal(red)  } |

## Περιγραφή σημείων συγχρονισμού με ψευτοκώδικα

|  |
| --- |
| (α) H επιθυμητη λειτουργια του mtx ειναι ο αμοιβαιος αποκλεισμος μεταξυ των αυτοκινητων ωστε να εκτελεσουν τον κρισιμο κωδικα τους με ασφαλεια. |
| (β)main:  init(mtx)  blue/red thread:  lock(mtx)  //krisimos kodikas  unlock(mtx) |

## Περιγραφή σημείων συγχρονισμού με ψευτοκώδικα

|  |
| --- |
| (α) Η επιθυμητη λειτουργια των mtx\_main και το cond\_main ειναι να μπλοκαρουν τη main μετα την δημιουργια των threads, μεχρι να περασουν ολα τα αυτοκινητα απο τη γεφυρα. Το τελευταιο αυτοκινητο ξεμπλοκαρει τη main. |
| (β)main:  init(cond\_main)  while(1){  //zhtame apo ton xrhsth to plhthos ton red/blue cars  if(ola\_ta\_blue\_cars>0){  lock(mtx\_blue)  }  else{  lock(mtx\_red)  }  //dhmioyrgoyme ta threads ton cars  if(ola\_ta\_blue\_cars>0){  wait(cond\_main,mtx\_blue)  unlock(mtx\_blue)  }  else{  wait(cond\_main,mtx\_red)  unlock(mtx\_red)  }  red/blue thread:  if(ola\_ta\_red\_cars==count\_r && ola\_ta\_blue\_cars==count\_b){  signal(cond\_main)  } |