Zadatak 6.4. Problem starog mosta

Stari most je uski most i stoga postavlja ograničenja na promet. Na njemu istovremeno smiju biti najviše tri automobila koja voze u istom smjeru. Simulirati automobile dretvom `Auto` koja obavlja niže navedene radnje. Napisati pseudokod monitorskih funkcija `Popni_se_na_most (smjer)` i `Siđi_s_mosta()`.

```
Dretva Auto(smjerA) // smjerA = 0 ili 1
{
    Popni_se_na_most(smjerA)
    prijeđi_most
    Siđi_s_mosta(smjerA)
}
```

Rješenje A – skica rješenja, početna ideja "u glavi"

```
m-funkcija Popni_se_na_most(smjerA)
{
    Uđi_u_monitor() //treba 1 monitor
    dok je (most pun ili auti na mostu voze se u drugom smjeru)
        Čekaj_u_redu_uvjeta()
    kreni preko mosta // promjena varijabli stanja sustava
    Izađi_iz_monitora()
}

m-funkcija Siđi_s_mosta(smjerA)
{
    Uđi_u_monitor()
    odvezi_se_s_mosta  // promjena varijabli stanja sustava
    odblokiraj dretve koje čekaju a mogu prijeći
    Izađi_iz_monitora()
}
```

Mi ipak tražimo konkretno rješenje koje se može prepisati u neki programski jezik (npr. C).

Rješenje B – konkretno rješenje

Prvo: stanje sustava definirati potrebnim varijablama

Varijable i početne vrijednosti:

- `autiM = 0` – broj automobila na mostu
- `smjerM = 0` – 0/1 smjerovi (lijevo/desno), -1 - most je prazan
- `m` – monitor s redom uvjeta `red`

```
m-funkcija Popni_se_na_most(smjerA)
{
    Uđi_u_monitor(m)
    dok je (smjerM == 1 - smjerA ILI autiM == 3)
     // ili (autiM > 0 && smjerA != smjerM) || (autiM == 3))
        Čekaj_u_redu_uvjeta(m, red)
    smjerM = smjerA
    autiM++
    Izađi_iz_monitora(m)
}
m-funkcija Siđi_s_mosta(smjerA)
{
    Uđi_u_monitor(m)
    autiM--
    ako je autiM == 0 onda
        smjerM = -1
    Oslobodi_sve_iz_reda_uvjeta(red)
    Izađi_iz_monitora(m)
}
```

Komentari

- ovo je jednostavno i kratko rješenje
- tipične monitorske funkcije: jedna služi za zauzimanje sredstava (“most”), a druga za oslobađanje
- bilo bi efikasnije kada bismo pratili da li netko čeka i tek tada pozvali signal/broadcast jezgrine funkcije!
- Postoji problem izgladnjivanja – kolona s jedne strane može beskonačno čekati na kolonu s druge strane koja je prije počela s prelaskom mosta – općenito rješenje ovakvih problema ne postoji, treba napraviti kompromis (kao i kod problema pet filozofa)

Rješenje D – ublažen problem izgladnjivanja

Ideja: u slučaju "gužve" naizmjence propuštati po `N` automobila (npr. `N=10`).

Varijable:

- `m` – monitor
- `red[2]` – red za svaki smjer
- `autiM` – broj automobila na mostu (od 0 do 3)
- `smjerM` – 0/1 smjerovi, -1 - most je prazan
- `čeka[2]` – koliko automobila čeka s pojedine strane (početno 0)
- `krenulo` – koliko je auta krenulo preko mosta u sadašnjem smjeru – toliko će ih preći preko mosta u sadašnjem smjeru (početno 0)

```
m-funkcija Popni_se_na_most(smjerA) {
    smjerB = 1 - smjerA
    Uđi_u_monitor(m)
    čeka[smjerA]++
    dok je (autiM==3 || smjerM==smjerB || (čeka[smjerB]>0 && krenulo>= N))
        Čekaj_u_redu_uvjeta(m, red[smjerA])
    čeka[smjerA]--
    smjerM = smjer
    autiM++
    krenulo++
    Izađi_iz_monitora(m)
}
m-funkcija Siđi_s_mosta(smjerA) {
    smjerB = 1 - smjerA
    Uđi_u_monitor(m)
    autiM--
    ako je (autiM > 0) { //još ima auta na mostu
        ako je (čeka[smjerA]>0 && (čeka[smjerB]==0 || krenulo<N))
            Oslobodi_iz_reda_uvjeta(red[smjerA])
    }
    inače { //most prazan
        krenulo = 0
        ako je (čeka[smjerB] > 0) {
            smjerM = smjerB //pusti one iz suprotna smjera
            ako je (čeka[smjerB] <= 3) {
                Oslobodi_sve_iz_reda_uvjeta(red[smjerB], m)
            }
            inače {
                za i=1 do 3
                    Oslobodi_iz_reda_uvjeta(red[smjerB], m)
            }
        }
        inače {
            smjerM = -1 //svi mogu
        }
    }
    Izađi_iz_monitora(m)
}
```

Zadatak 6.5. Ping-pong dretve

Simulirati rad dretvi *ping* i dretvi *pong*. Dretve se nasumično pojavljuju u sustavu (stvaraju ih neke druge dretve) i u svom radu samo ispisuju poruku: dretve *ping* ispisuju `ping` dok dretve *pong* ispisuju `pong`. Sinkronizirati rad dretvi tako da:

a) ispis bude pojedinačan (unutar kritičnog odsječka)

b) dretve *ping* i *pong* naizmjence ispisuju poruke (ispis: `ping pong ping pong ...`)

c) dretve *ping* i *pong* ispisuju poruke tako da se uvijek pojavljuju dva `ping`-a prije svakog `pong`-a (ispis: `ping ping pong ping ping pong ...`)

d) dretve *ping* i *pong* ispisuju poruke tako da se uvijek pojavljuju barem dva `ping`-a prije svakog `pong`-a (ispis: `ping ping ping pong ping ping pong ping ...`)

Rješenje za a) pojedinačan ispis

```
dretva ping()
{
    Čekaj_BSEM(1)
    Ispiši("ping")
    Postavi_BSEM(1)
}
```

```
dretva pong()
{
    Čekaj_BSEM(1)
    Ispiši("pong")
    Postavi_BSEM(1)
}
```

Rješenje za b) naizmjeničan ispis

```
dretva ping()
{
    Čekaj_BSEM(1)
    Ispiši("ping")
    Postavi_BSEM(2)
}
```

```
dretva pong()
{
    Čekaj_BSEM(2)
    Ispiši("pong")
    Postavi_BSEM(1)
}
```

```
dretva ping()
{
    Uđi_u_monitor(m)
    dok je na_redu != PING
        Čekaj_u_redu_uvjeta(red1, m)
    Ispiši("ping")
    na_redu = PONG
    Oslobodi_iz_reda_uvjeta(red2)
    Izađi_iz_monitora(m)
}
```

```
dretva pong()
{
    Uđi_u_monitor(m)
    dok je na_redu != PONG
        Čekaj_u_redu_uvjeta(red2, m)
    Ispiši("pong")
    na_redu = PING
    Oslobodi_iz_reda_uvjeta(red1)
    Izađi_iz_monitora(m)
}
```

Rješenje za c) dva PING-a prije svakog PONG-a (PING PING PONG PING PING PONG . . .)

i) rješenje s općim semaforima

```
dretva ping()
{
    Čekaj_OSEM(1)
    Čekaj_BSEM(1)
    Ispiši("ping")
    Postavi_BSEM(1)
    Postavi_OSEM(2)
}
```

```
dretva pong()
{
    Čekaj_BSEM(2)
    Čekaj_OSEM(2)
    Čekaj_OSEM(2)
    Ispiši("pong")
    Postavi_OSEM(1)
    Postavi_OSEM(1)
    Postavi_BSEM(2)
}
```

ii) rješenje s varijablom

```
dretva ping()
{
    Čekaj_BSEM(1)
    Ispiši("ping")
    brojač++
    ako je brojač < 2 tada
        Postavi_BSEM(1)
    inače
        brojač = 0
        Postavi_BSEM(2)
}
```

```
dretva pong()
{
    Čekaj_BSEM(2)
    Ispiši("pong")
    Postavi_BSEM(1)
}

Početne vrijednosti:
brojač = 0
BSEM[1].v = 1
BSEM[2].v = 0
```

Rješenje za d) barem dva PING-a prije svakog PONG-a

Potrebno:
`BSEM[KO]` - kritični odsječak, početna vrijednost 1
`BSEM[PONG]` - čekaju dretve pong, početna vrijednost 0
`brojač` - koliko puta je ipisan ping između dva ponga, početna vrijednost 0

```
dretva ping()
{
    Čekaj_BSEM(KO) //Uđi*

    ispiši("ping")
    brojač++
    ako je brojač > 1
        Postavi_BSEM(PONG) //Oslobodi*

    Postavi_BSEM(KO) //Izađi*
}
```

```
dretva pong()
{
    Čekaj_BSEM(KO) //Uđi*

    dok je (brojač < 2) {
        Postavi_BSEM(KO) //(ništa)
        Čekaj_BSEM(PONG) //Čekaj*
        Čekaj_BSEM(KO)   //(ništa)
    }
    ispiši("pong")
    brojač = 0

    Postavi_BSEM(KO) //Izađi*
}
```

Zadatak 6.11. Kratki i dugi poslovi

U nekom sustavu postoje dva tipa dretvi: jedna dretva `prihvati` i `N` dretvi `obradi`. Dretva `prihvati` čeka i zaprima nove poslove sa: `posao = dohvati_idući()`. Posao može biti kratki ili dugi (`posao.tip == KRATKI/DUGI`). Kratke poslove dretva `prihvati` stavlja u njihov red sa `stavi_kratki(posao)`, a duge u njihov sa `stavi_dugi(posao)`. Dretve `obradi` preko `posao=uzmi_kratki()/uzmi_dugi()` uzimaju posao iz jednog ili drugog reda te ga obrađuju s `obradi(posao)`. Dretva `obradi` treba odabrati dugi posao osim ako:

- nema dugih poslova u njihovu redu (a kratkih ima)
- ima više od 10 kratkih poslova u njihovu redu
- ima kratkih poslova u njihovu redu i trenutno barem pet dretvi već obrađuje duge poslove.

Napisati pseudokod za oba tipa dretvi korištenjem monitora za sinkronizaciju. Korištenje redova (operacije `stavi*` i `uzmi*`) zaštititi. Obrade različitih poslova od strane različitih dretvi moraju se moći obaviti i paralelno (izvan monitora).

Rješenje:

- `m` – monitor
- `r` – red uvjeta u kojem čekaju radne dretve na novi posao
- `br_k` – broj kratkih poslova u redu
- `br_d` – broj dugih poslova u redu
- `n_d` – broj dretvi koje trenutno rade na dugim poslovima

```
dretva prihvati()
{
  ponavljaj {
    posao = dohvati_idući()
    uđi_u_monitor(m)
    ako je posao.tip == kratki {
      stavi_kratki(posao)
      br_k++
    }
    inače {
      stavi_dugi(posao)
      br_d++
    }
    oslobodi_iz_reda_uvjeta(r)
    izađi_iz_monitora(m)
  }
}
```

```
dretva obradi()
{
  uđi_u_monitor(m)
  ponavljaj {
    dok je (br_d==0 && br_k==0)
      čekaj_u_redu_uvjeta(r,m)

    ako je (br_d==0 && br_k>0) ||
    (br_k>10) || (br_k>0 && n_d>=5)
    {
      posao = uzmi_kratki()
      br_k--
    }
    inače {
      posao = uzmi_dugi()
      br_d--
      n_d++
    }
    izađi_iz_monitora(m)
    obradi(posao)
    uđi_u_monitor(m)
    ako je posao.tip == dugi
      n_d--
  }
}
```

Zadatak 6.12. H_2O

Sinkronizirati dretve vodika i kisika koje stvaraju molekulu vode H_2O. Dretve se stvaraju nasumično. Za stvaranje molekule koristi se "kalup". Kad sva tri atoma budu u kalupu molekula vode nastaje. Dretva završava s radom kad uđe u kalup, osim ako to nije ona koja treba "sastaviti molekulu" i izbaciti ju iz kalupa.

Prvo rješenje, uz zasebne dretve, gdje atom O upravlja sklapanjem

```
dretva O
{
    ČekajBSEM(O)
    uđi u kalup na mjesto O
    ČekajOSEM(HuK)
    ČekajOSEM(HuK)
    sklopi molekulu
    PostaviOSEM(H)
    PostaviOSEM(H)
    PostaviBSEM(O)
}
```

```
dretva H
{
    ČekajOSEM(H)
    uđi u kalup na mjesto H
    PostaviOSEM(HuK)
}

Početne vrijednosti:
OSEM[H].v = 2
BSEM[O].v = 1
OSEM[HuK].v = 0
```

Rješenje s monitorima:

```
Početne vrijednosti:
MAX[2] = {2,1} -- koliko atoma pojedina tipa ide u molekulu
br[2] = {0, 0} -- koliko atoma pojedina tipa se nalazi u kalupu
m - monitor
ulaz[2] - dva reda uvjeta (za H i za O)

dretva atom(X) //X je H(1) ili O(2)
{
    Uđi_u_monitor(m)
    dok je (br[X] == MAX[X])
        Čekaj_u_redu_uvjeta(m, ulaz[X])
    uđi u kalup na mjesto za X
    br[X]++
    ako je (br[H]+br[O] == 3) { //zadnji atom molekule H2O
        sklopi molekulu
        br[H] = 0
        br[O] = 0
        Oslobodi_iz_reda_uvjeta(ulaz[H])//pusti nove atome u kalup
        Oslobodi_iz_reda_uvjeta(ulaz[H])
        Oslobodi_iz_reda_uvjeta(ulaz[O])
    }
    //inače: atom je u kalupu, dretva može izaći iz monitora i završiti
    Izađi_iz_monitora(m)
}
```