Zadatak 5.2. Problem pušača cigareta

(Ograda – ne reklamiramo cigarete (naprotiv), ali to je ime algoritma – Patil, 1971.)

Zamislimo sustav s tri dretve pušača i jednom dretvom trgovcem. Svaki pušač neprestano savija cigarete i puši. Kako bi se savila i popušila cigareta potrebno je imati tri sastojka: duhan, papir i šibice. Jedan pušač ima u neograničenim količinama samo papir, drugi samo duhan, a treći samo šibice. Trgovac ima sva tri sastojka u neograničenim količinama. Trgovac nasumice stavlja na stol dva različita sastojka. Pušač koji ima treći sastojak uzima sastojke sa stola, signalizira trgovcu, savija cigaretu i puši. Trgovac stavlja nova dva sastojka na stol i ciklus se ponavlja. Na početku je stol prazan. Napisati dretve pušača i trgovca tako da se one međusobno ispravno sinkroniziraju s pomoću dva semafora. Napisati početne vrijednosti semafora.

Početno rješenje:

```
dretva Trgovac ()
{
   ponavljaj
   {
      (s1, s2) = nasumice odaberi dva različita sastojka

      Čekaj_BSEM ( stol_prazan )
      stavi_sastojke_na_stol ( s1, s2 )   // KO
      Postavi_BSEM ( nešto_ima )
   }
   do zauvijek
}

dretva Pušač(p)
{
   //svaki pušač ima različiti 'p', npr. 1,2,3
   (r1, r2) = sastojci_koje_pušač_nema (p)

   ponavljaj
   {
      Čekaj_BSEM ( nešto_ima )
      ako ( na_stolu_sastojci ( r1, r2 ) = DA )     // KO
      {
         uzmi_sastojke ( r1, r2 )                   // KO
         Postavi_BSEM ( stol_prazan )
         napravi cigaretu ...
      }
      inače {
         Postavi_BSEM ( nešto_ima )
      }
   } do zauvijek
}
```

Početne vrijednosti: `BSem[stol_prazan] = 1`, `BSem[nešto_ima] = 0`.

Komentar:
Može se pojaviti radno čekanje: dok jedan ili dva pušača puše, trgovac može na stol staviti sastojke upravo za tog pušača koji već puši – drugi pušači će za to vrijeme ulaziti/izlaziti iz KO – vrtiti se u petlji – neće čekati.

Drugo rješenje s 5 semafora: `stol_prazan`, `KO` i po jedan za svakog pušača; Trgovac postavlja sva tri!

```
dretva Trgovac()
{
   ponavljaj
   {
      (s1, s2) = nasumice odaberi dva različita sastojka
      Čekaj_BSEM(stol_prazan)
      Čekaj_BSEM(KO)
      stavi_sastojke_na_stol(s1, s2)
      Postavi_BSEM(KO)
      Postavi_BSEM(1)
      Postavi_BSEM(2)
      Postavi_BSEM(3)
   }
   do ZAUVIJEK
}

dretva Pušač(p)  //p je iz skupa {1,2,3}
{
   (r1, r2) = sastojci_koje_pušač_nema(p)
   ponavljaj
   {
      Čekaj_BSEM(p)
      Čekaj_BSEM(KO)
      ako (na_stolu_sastojci(r1, r2) == DA )
      {
         uzmi_sastojke(r1, r2)
         Postavi_BSEM(KO)
         Postavi_BSEM(stol_prazan)
         smotaj, zapali i puši
      }
      inače {
         Postavi_BSEM(KO)
      }
   }
   do ZAUVIJEK
}
```

Početne vrijednosti: `BSem[stol_prazan] = 1`, `BSem[KO] = 1`, `BSem[p] = 0`.

Problem – puno semafora (5); što ako nema `KO`? (paralelno gledaju, netko uzima, trgovac stavlja . . .)

Zadatak 5.2.x

Sličan zadatak Zadatku 5.2. ali „zdraviji“ (možda).

U neku trgovinu ulaze kupci koje si žele napraviti sendvič. Za napraviti sendvič te osobe trebaju kruh, šunku i sir. Svaka osoba već ima jedan sastojak i trebaju joj iduća dva. Trgovac u trgovini ima sva tri sastojka u velikoj količini. Trgovac nasumice odabire dva različita sastojka i stavlja ih na stol. Kupac koji je u trgovini i koji ima različit sastojak od onih na stolu, uzima sastojke sa stola, izlazi iz trgovine, slaže si sendvič i jede. U trgovini u svakom trenutku može biti najviše tri kupca i to s različitim sastojcima. Tek kad neki kupac izađe, u trgovinu može ući drugi, ali samo takav koji treba isti sastojak kao i ovaj koji izlazi. Riješiti sinkronizaciju semaforima.

Početno rješenje:

- ulaz u trgovinu štititi s tri binarna semafora: `ulaz_kruh`, `ulaz_šunka` i `ulaz_sir`
- signal trgovcu da je stol prazan preko binarnog semafora: `stol_prazan`
- za svaki sastojak na stolu po jedan binarni semafor: `kruh, šunka, sir`

Početne vrijednosti semafora:

```
BSEM[ulaz_kruh].v = BSEM[ulaz_šunka].v = BSEM[ulaz_sir] = BSEM[stol_prazan] = 1
BSEM[kruh].v = BSEM[šunka].v = BSEM[sir] = 0
```

```
dretva trgovac
{
  ponavljaj {
    ČekajBSEM(stol_prazan)
    s1,s2 = nasumice odaberi dva različita sastojka //dva od: kruh, šunka, sir
    PostaviBSEM(s1)
    PostaviBSEM(s2)
  }
}
```

```
dretve osoba_s_kruhom
{
  ČekajBSEM(ulaz_kruh)
  uđi u trgovinu

  ČekajBSEM(šunka)
  ČekajBSEM(sir)
  uzmi sastojke sa stola
  PostaviBSEM(stol_prazan)

  odi van trgovine
  PostaviBSEM(ulaz_kruh)
  napravi sendvič
  jedi
}
```

```
dretve osoba_sa_šunkom
{
  ČekajBSEM(ulaz_šunka)
  uđi u trgovinu

  ČekajBSEM(kruh)
  ČekajBSEM(sir)
  uzmi sastojke sa stola
  PostaviBSEM(stol_prazan)

  odi van trgovine
  PostaviBSEM(ulaz_šunka)
  napravi sendvič
  jedi
}
```

```
dretve osoba_sa_sirom
{
  ČekajBSEM(ulaz_sir)
  uđi u trgovinu

  ČekajBSEM(kruh)
  ČekajBSEM(šunka)
  uzmi sastojke sa stola
  PostaviBSEM(stol_prazan)

  odi van trgovine
  PostaviBSEM(ulaz_sir)
  napravi sendvič
  jedi
}
```

Problem prikazanog rješenja: može se dogoditi *potpuni zastoj!*
Dva različita kupca mogu uzeti različite sastojke (proći samo prvi od crvenih semafora), ali ni jedan neće imati sva tri te neće dojaviti trgovcu da je stol prazan. Trgovac neće staviti nove sastojke i svi će čekati beskonačno.
Problem potpuna zastoja će detaljnije biti obrazložen u okviru 6. poglavlja.

Potpuni zastoj nije dozvoljen u ispranom rješenju kada se koriste semafori (i monitori)!

Rješenje s provjerama

- ulaz u trgovinu štititi s tri binarna semafora: `ulaz_kruh`, `ulaz_šunka` i `ulaz_sir`
- signal trgovcu da je stol prazan preko binarnog semafora: `stol_prazan`
- signal kupcima da su na stolu sastojci: `stol_pun`
- zajedničke varijable: `s1`, `s2` – označavaju sastojke koji su na stolu

Početne vrijednosti semafora:
`BSEM[ulaz_kruh].v = BSEM[ulaz_šunka].v = BSEM[ulaz_sir] = BSEM[stol_prazan] = 1`
`BSEM[stol_pun].v = 0`

```
dretva trgovac
{
  ponavljaj {
    ČekajBSEM(stol_prazan)
    s1,s2 = nasumice odaberi dva različita sastojka() //dva od: kruh, šunka, sir
    PostaviBSEM(stol_pun)
  }
}

dretve osoba_s_kruhom
{
  ČekajBSEM(ulaz_kruh)
  uđi u trgovinu

  ponavljaj {
    imam_sve = LAŽ              //imam_sve: lokalna varijabla, svaka dretva ima svoju
    ČekajBSEM(stol_pun)
    ako je ( s1 != "kruh" I s2 != "kruh" ) onda {
      uzmi sastojke sa stola
      imam_sve = ISTINA
      PostaviBSEM(stol_prazan)
    }
    inače {
      PostaviBSEM(stol_pun)
    }
  }
  dok je imam_sve == LAŽ       //ponavljaj petlju dok je uvjet ispunjen

  odi van trgovine
  PostaviBSEM(ulaz_kruh)
  napravi sendvič
  jedi
}
```

Sličan kod je za **osoba_sa_šunkom** i **osoba_sa_sirom**, uz odgovarajuće zamjene (obojani tekst).

Problem: radno čekanje
Primjerice, ukoliko trgovac na stol stavi sastojke za kupca koji nije u trgovini, ostala dva kupca (ili i samo jedan) će svejedno izvoditi dok je petlju!

Radno čekanje nije dozvoljeno u ispranom rješenju kada se koriste semafori (i monitori)!

Rješenje s više semafora

- ulaz u trgovinu štititi s tri binarna semafora: `ulaz_kruh`, `ulaz_šunka` i `ulaz_sir`
- signal trgovcu da je stol prazan preko binarnog semafora: `stol_prazan`
- signal kupcima da su na stolu sastojci: `kruh, šunka, sir`
- zajedničke varijable: `s1`, `s2` – označavaju sastojke koji su na stolu
- korištenje zajedničkih varijabli zaštićeno kritičnim odsječkom, semaforom `KO`

Početne vrijednosti semafora:

```
BSEM[ulaz_kruh].v = BSEM[ulaz_šunka].v = BSEM[ulaz_sir] = BSEM[stol_prazan] = 1
BSEM[KO].v = 1
BSEM[kruh].v = BSEM[šunka].v = BSEM[sir] = 0
```

```
dretva trgovac
{
  ponavljaj {
    ČekajBSEM(stol_prazan)

    ČekajBSEM(KO)
    s1,s2 = nasumice odaberi dva različita sastojka() //dva od: kruh, šunka, sir
    PostaviBSEM(KO)

    PostaviBSEM(kruh)        // svaki puta kad se nešto stavi na stol
    PostaviBSEM(šunka)       // samo se jednom javi „svakom kupcu“
    PostaviBSEM(sir)
  }
}
```

```
dretve osoba_s_kruhom
{
  ČekajBSEM(ulaz_kruh)
  uđi u trgovinu

  ponavljaj {
    imam_sve = LAŽ             //imam_sve: lokalna varijabla, svaka dretva ima svoju
    ČekajBSEM(kruh)
    ČekajBSEM(KO)
    ako je ( s1 != "kruh" I s2 != "kruh" ) onda {
      uzmi sastojke sa stola
      imam_sve = ISTINA
      PostaviBSEM(stol_prazan)
    }
    PostaviBSEM(KO)
  }
  dok je imam_sve == LAŽ       //ponavljaj petlju dok je uvjet ispunjen

  odi van trgovine
  PostaviBSEM(ulaz_kruh)
  napravi sendvič
  jedi
}
```

Sličan kod je za **`osoba_sa_šunkom`** i **`osoba_sa_sirom`**, uz odgovarajuće zamjene (obojani tekst).

Algoritam radi. Ali treba 5 semafora (+3 za kontrolu ulaska u trgovinu)!

Zadatak 5.3. Prioritetni redovi i semafori (kraći)

U jednoprocesorskom računalu pokrenut je sustav dretvi D_1, D_2 i D_3 s prioritetima 1, 2 i 3 respektivno. Najviši prioritet je 3. Svi zadaci koje obavljaju dretve su istog oblika D_x. Red pripravnih dretvi i red semafora su prioritetni. Aktivna je dretva koja je prva u redu pripravnih (nema posebnog reda aktivnih dretvi). Prije pokretanja sustava dretvi semafor `S` je bio zatvoren. Nakon nekog vremena (kad se navedene dretve nađu u redu semafora S), pokrene se dretva G. Što će se ispisati na zaslonu?

```
Dretva Dx{
    dok je(1){
        Čekaj_BSEM(S)
        piši(Px)        a
        Postavi_BSEM(S)-------------
        piši(Zx)        b
    }
}
dretva G {
    Postavi_BSEM(S)
}
```

Podsjetnik na jezgrine funkcije:

```
j-funkcija ČEKAJ_BSEM (S)
{
    pohrani kontekst u opisnik Aktivna_D

    ako ( BSEM[S].v == 1 ) {
        BSEM[S].v = 0
    }
    inače {
        stavi_u_red ( makni_prvu_iz_reda ( Aktivna_D ), BSEM[S] )
        stavi_u_red ( makni_prvu_iz_reda ( Pripravne_D ), Aktivna_D )
    }

    obnovi kontekst iz opisnika Aktivna_D
}
j-funkcija POSTAVI_BSEM (S)
{
    pohrani kontekst u opisnik Aktivna_D

    ako ( red BSEM[S] nije prazan ) {
        stavi_u_red ( makni_prvu_iz_reda ( Aktivna_D ), Pripravne_D )
        stavi_u_red ( makni_prvu_iz_reda ( BSEM[S] ), Pripravne_D )
        stavi_u_red ( makni_prvu_iz_reda ( Pripravne_D ), Aktivna_D )
    }
    inače {
        BSEM[S].v = 1
    }

    obnovi kontekst iz opisnika Aktivna_D
}
```

Rješenje (tablica ima 15-tak redaka)

BSEM(S).red	.v	Pripravne_D	P	Z	
-	0	3 2 1			(3) `Čekaj_BSEM(S)`
3	0	2 1			(2) `Čekaj_BSEM(S)`
3 2	0	1			(1) `Čekaj_BSEM(S)`
3 2 1	0	–			
3 2 1	0	–			(G) `Postavi_BSEM(S)`
2 1	0	3a	3		(3) `piši(P3); Postavi_BSEM(S)`
1	0	3b 2a		3	(3) `piši(Z3); Čekaj_BSEM(S)`
3 1	0	2a	2		(2) `piši(P2); Postavi_BSEM(S)`
1	0	3a 2b	3		(3) `piši(P3); Postavi_BSEM(S)`
–	0	3b 2b 1a		3	
3	0	2b 1a		2	
3 2	0	1a	1		
2	0	3a 1b	3		
–	0	3b 2a 1b		3	
3	0	2a 1b	2		
–	0	3a 2b 1b	3		ponavlja se; stacionarno stanje
–	1	3b 2b 1b		3	
–	0	3a 2b 1b	3		

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)
}
```

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*
}
```