OS ZI 2012./2013. Zadaci sa riješenjima

1. U sustavu sa straničenjem, pri izvođenju nekog programa koji zauzima 6 stranica, javljaju se redom sljedeći zahtjevi za stranicama: *2, 1, 2, 3, 5, 3, 4, 5, 6, 3, 2, 6.* Za program je na raspolaganju 3 okvira.

a) Koji je sadržaj okvira nakon 4. zahtjeva?

Rj: Imamo 3 okvira. Budući da je 3. zahtjev isti kao i prvi, nebitno je koju strategiju koristimo, budući da se neće dogoditi niti jedno izbacivanje. Po koracima to izgleda ovako:

Zahtjev	O1	O2	O3
2	2		
1	2	1	
2	2	1	
3	2	1	3

Odgovor je **213**

b) Koji je sadržaj okvira nakon zadnjeg zahtjeva, uz optimalnu strategiju zamjene stranica (redoslijed nije bitan) ?

Rj: Optimalna strategija funkcionira ovako: Ukoliko moramo izbaciti neku stranicu, izbacujemo onu koja će se koristiti najdalje u budućnosti. Možemo to učiniti i ovako: Kad smo na koraku **n** i prisiljeni smo izbaciti neku stranicu, za svaku stranicu gledamo broj koraka u kojem se opet pojavljuje **p**. Taj će se broj opet koristiti za **t = p – n** koraka. Ukoliko se zahtjevi za tom stranicom više ne pojavljuju, uzimamo **t** = ∞. Zatim gledamo za koji broj je **t** maksimalan i taj broj izbacujemo. U praksi:

Zahtjev	O1	O2	O3	Stanje
2	2			miss
1	2	1		miss
2	2	1		hit
3	2	1	3	miss
5	2	5	3	miss
3	2	5	3	hit
4	4	5	3	miss
5	4	5	3	hit
6	6	5	3	miss
3	6	5	3	hit
2	6	2	3	miss
6	6	2	3	hit

Plavom bojom označeni su koraci u kojima smo morali koristiti strategiju zamjene stranica. U 5. koraku, kad se pojavi zahtjev 5, postupak je jednostavak. Stranice 2 i 3 opet će se tražiti, dok se stranica 1 više neće tražiti, dakle **t**(1) = ∞. Stoga izbacujemo stranicu 1.

U koraku 7, pri pojavi zahtjeva 4, takvog slučaja nema. Kad pogledamo ponovne zahtjeve za stranice 2, 3 i 5 vidimo **t**(2) = 4, **t**(3) = 3, **t**(5) = 1. Stoga izbacujemo stranicu 2, jer je njezin **t** najveći.

U koraku 9, situacija je slična koraku 5. **t**(4) = ∞, **t**(5) = ∞, **t**(3) = 1. Dakle, možemo izbaciti 4 i tako dobiti sustav **653** ili izbaciti 5 i dobiti sustav **463**.

Konačno, u koraku 11, uzimamo u obzir dvije situacije:
a) Naš sustav je **653**, kao i u tablici.

t(6) = 1, **t**(5) = **t**(3) = ∞. Izbacivanjem jednog od njih dobivamo **623** ili **652**.

b) Naš sustav je **463**, nije prikazano u tablici.

t(6) = 1, **t**(4) = **t**(3) = ∞. Izbacivanjem 3 ili 4 dobivamo **623** ili **642**.

Unijom tih riješenja dobivamo 3 konačna točna riješenja: **623**, **642** i **652**.

Od tih riješenja ponuđeno je riješenje **623**, dakle to zaokružujemo.

TIP: Do sad je uvijek u sustavima sa optimalnom strategijom bilo moguće doći i jednostavnije, bez provjeravanja svih mogućih riješenja: Kad **t** bude jednak za više brojeva, tj. zahtjevi budu jednako daleko u budućnosti, zamjenite prvi od njih u tablici. To je do sad uvijek bio ponuđeni točan odgovor.

c) Koliko promašaja uzrokuje optimalna strategija?

Rj: Odgovor je 7, vidi tablicu iz iz b) dijela zadatka.

d) Koliki je broj promašaja uz satni mehanizam (clock)?

Rj: Prije svega **ne mogu garantirati** točnost postupka. Koristio sam postupak koji sam našao na internetu i koji odgovara njihovoj simulaciji na službenoj stranici OS-a, ali ne daje točan odgovor u e) zadatku (moguće da je njihova greška).

Dakle, satni mehanizam funkcionira ovako:

```
funkcija Satni {
        dok( okvir nije odabran ) {
                ako(okvir.A == 0)
                        zamjeni sadržaj;
                inače
                        okvir.A = 0;
                pomakni kazaljku na idući okvir;
        }
}
```

Konstruirajmo tablicu za zadani set zahtjeva:

Zahtjev	O1	O2	O3	Zastavice	Stanje
2	2			**1**00	miss
1	2	1		**1**10	miss
2	2	1		**1**10	hit
3	2	1	3	**1**11	miss
5	5	1	3	1**0**0	miss
3	5	1	3	1**0**1	hit
4	5	4	3	11**1**	miss
5	5	4	3	11**1**	hit
6	5	4	6	**0**01	miss
3	3	4	6	1**0**1	miss
2	3	2	6	11**1**	miss
6	3	2	6	11**1**	hit

Boldani bit označava trenutnu poziciju kazaljke. Plavi redovi označavaju redove u kojima smo koristili satni mehanizam za odabir stranice koju ćemo izbaciti.

Kao što vidimo s tom strategijom imamo **8** promašaja, što je i točan odgovor.

e) Koje su vrijednosti bita pristupa (A) stranica u okvirima na kraju primjene satnog mehanizma?

Rj: Pogledajmo tablicu iz d) dijela zadatka. Sudeći po njoj, kao i po simulaciji na službenoj stranici OS-a (na fer.unizg.hr), točan odgovor bio bi **111**. U službenim je riješenjima odgovor je **011**.

2. U sustavu sa straničenjem ukupan broj stranica logičkog adresnog prostora je 8, a fizički adresni prostor sadrži jedank broj okvira. Veličina jedne stranice je 8 riječi (8 različitih adresa). Za zadano trenutno stranje tablice prevođenja nekog procesa, za svaku logičku adresu odredite odgovarajuću fizičku adresu, odnosno uvjet za nastanak prekida.

	okvir	prisutnost		okvir	prisutnost
0	100	1	4	101	0
1	011	0	5	010	1
2	111	1	6	001	1
3	110	1	7	010	0

a) LA: 101101; FA = ?

Rj: Ovaj zadatak je u biti dosta jednostavan, ali zahtjeva malo znanja teorije. Dakle, iz logičke adrese možemo isčitati dva dijela: Broj stranice i adresu riječi unutar stranice. Budući da i jednih i drugih ima 8, to znači da svaki dio zauzima 3 bita. Drugim riječima, prva 3 bita u logičkoj adresi su broj stranice, a druga tri bita su adresa riječi unutar stranice.

Preslikavanje funkcionira ovako: Uzimamo gornja tri bita LA, to je naš broj stranice. Prebacimo to u dekadski broj i pogledajmo u tablicu. **Ako je bit prisutnosti 0, stranica nije u memoriji!** U tom slučaju ne računamo fizičku adresu, već se događa prekid. Ukoliko je bit prisutnosti 1, isčitamo broj okvira. Fizička adresa ima istu poziciju unutar stranice, dakle donja 3 bita koja samo prepišemo, ali za gornja tri bita uzimamo broj okvira u kojoj je stranica, a ne broj stranice. Znači:

(LA) broj stranice|adresa riječi → (FA) broj okvira|adresa riječi

Konkretno 1011101 → 101|101 → Stranica 5, riječ 5
Za stranicu 5 postavljen je bit prisutnosti i adresa okvira je 010, tj 2
Dakle, FA = 010|101 = **010101**

b) LA: 011111; FA = ?

Rj:
011|111 → Stranica 3, riječ 7.
Za stranicu 3 postavljen je bit prisutnost i broj okvira je 110.
FA = 110|111 = **110111**

c) LA: 100010; FA = ?;

Rj:
100|010 → Stranica 4, riječ 2
Za stranicu 4 bit prisutnosti je **0** i **događa se prekid** (zahtjev za stranicom, straničenje, bla, bla).

3. Na neki poslužitelj dolazi prosječno 100 zahtjeva u minuti, dok ih poslužitelj može obraditi 300 u minuti. Na isti poslužitelj tada se propuste i drugi zahtjevi, kojih dolazi prosječno 80 u minuti. Koje bi prosječno trajanje zadržavanja bilo kad bi poslužitelj obrađivao samo druge zahtjeve, ako je prosječno zadržavanje za obje skupine zahtjeva 1 sekundu? Dolazak zahtjeva podliježe Poissonovoj, a trajanje obrade eksponencijalnoj razdiobi?

a) Opterećenje poslužitelja uz obje skupine zahtjeva?

Rj: Kad prebacimo sve u sekunde, vrijedi $\alpha1 = 1.67$, $\alpha2 = 1.33$, $\beta1 = 5$, $T(oba) = 1$
Znamo da vrijedi i $\mathbf{\alpha = \alpha1 + \alpha2 = 3}$

Kod Poisson-exp. raspodijele vrijedi izraz:
$T = 1/(\beta - \alpha) \rightarrow \beta = 1/T + \alpha$
Kako znamo ukupni α, kao i T za oba posla, dobivamo $\mathbf{\beta = 4}$

Konačno, $\mathbf{\rho = \alpha/\beta = 0.75}$
To je naše (točno) riješenje.

b) Opterećenje samo uz drugu skupinu zahtjeva:

Rj: Znamo da vrijedi $\rho = \rho1 + \rho2$
Imamo ukupni ρ iz a) dijela zadatka, a imamo i $\alpha1$ i $\beta1$.
Stoga, $\mathbf{\rho1 = \alpha1/\beta1 = 0.334}$

Iz toga dobivamo $\mathbf{\rho2 = \rho - \rho1 \approx 0.416}$
Kako sam zaokruživao, ne dobivam 100% egzaktan rezultat, ali točan odgovor je B, koji se razlikuje tek u 3. decimali. (5/12)

c) Prosječno vrijeme zadržavanja samo za drugu skupinu poslova?

Već znamo da je $T2 = 1/(\beta2 - \alpha2)$.
Prosječan broj dolazaka imamo, ali fali nam $\beta2$. Srećom, $\rho2 = \alpha2/\beta2 \rightarrow \beta2 = \alpha2/\rho2 \approx 3.2$
Uvrštavanjem dobivamo $T2 \approx 0.53$ s $= 0.0089$ min
To približno odgovara odgovoru C, 1/112 min (do na 5. decimalu)

4. U redu pripravnih dretvi nalaze se dretve D1, D2 i D3, tim redoslijedom (D1 je prva u redu). Trajanje izvođenja dretve D1 je 5 ms, dretve D2 3 ms, te dretve D3 4 ms. Sustav koristi posluživanje redom prispjeća i kružno posluživanje uz trajanje kvanta 1ms.

a) Koji su trenuci završetka dretvi ako se izvode redom prispjeća? (Odgovor oblika D1, D2, D3)

Rj: Ovaj zadatak najlakše je riješiti grafom (y os – broj poslova, x os – vrijeme / ms):

D3	-	-
D2	D3	-
D1	D2	D3

0ms 5ms 8ms 12ms

Svi poslovi su prisutni u trenutku t = 0ms, a prvi počinje D1, kako je zadano. On traje 5ms pa završava u 0+5 = 5ms. Kada je on gotov počinje D2, koji završava u 5 + 3 = 8ms, te konačno D3 u 8 + 4 = 12 ms

Odgovor je stoga **5, 8, 12**

b) Koji je prosječni broj poslova (dretvi) u sustavu u periodu dok sve dretve ne završe?

Rj: Kod ovakvog deterministčkog sustava **n** (prosječni broj poslova) je lako odrediti.
Vrijedi **n = S/t** gdje je **S** površina ispod grafa. **t** je u ovom slučaju 12 ms.

n = [(5*3) + (3*2) + (4*1)]/12 = **25/12**

c) Koji su trenutci završetka dretvi ako se koristi kružno posluživanje? (Odgovor oblika D1, D2, D3)

Rj: Kod kružnog posluživanja poslovi se (u općem slučaju) obavljaju jedan kvant, nakon čega se vraćaju u kraj reda. Dakle, poslovi će se obavljati redom D1 → D2 → D3 → D1 → …

Grafički to izgleda ovako (stupac tablice je 1ms, plava boja označava početak, crvena kraj posla):

D1	D2	D3	D1	D2	D3	D1	D2	D3	D1	D3	D1

0ms 1ms 2ms 3ms 4ms 5ms 6ms 7ms 8ms 9ms 10ms 11ms 12

Iz grafa lagano iščitavamo da je riješenje **12, 11, 8**

d) Koje je prosječno trajanje zadržavanja dretvi uz kružno posluživanje?

Rj: Također možemo iščitati iz tablice. Dok prvi posao ne završi u sustavu imamo 3 posla, između tog trenutka 2, te konačno 1. Opet vrijedi **n = S/t**, **t** = 12 ms.

n = [((8 - 0)*3) + ((11 – 8) * 2) + ((12-11) * 1)]/12 = 31/12

5. U jednoprocesorskom računalu pokrenut je sustav dretvi D1, D2, D3, D4. Indeks dretve predstavlja i prioritet dretve, a najviši je prioritet 1. Svi zadaci koje obavljaju dretve su istog oblika Dx. 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 dretve semafor S bio je neprolazan. Nakon nekog vremena sve dretve se nađu u redu semafora S (prilikom prvog poziva ČekajBSEM(S)). U tom stanju pozove se procedura PostaviBSEM(S).
Tablično prikažite (dodatni papir) promjene stanja sustava dretvi:

```
Dretva Dx {
        ČekajBSEM(S);
        piši(Ax);
        PostaviBSEM(S);
        piši(Bx);
        ČekajBSEM(S);
        piši(Cx);
        PostaviBSEM(S);
}
```

a) Prvih 5 ispisa sustava:

Rj: Konstuirajmo odmah cijelu tablicu. Ovaj tip zadataka nije posebno zahtjevan, ali treba jako paziti pri svakom koraku, budući da se u simulaciji događa puno toga. Aktivni thread je označen boldom.

S.v	Red Semafora	Red Pripravnih	Ispis	Komentar
1	D1, D2, D3, D4	-	-	Nešto postavlja S
0	D2, D3, D4	**D1**	A1	D1 počinje, ispisuje
1	D2, D3, D4	**D1**	-	D1 postavlja S
0	D3, D4	**D1, D2**	B1	D1 ima veći prioritet
0	D1, D3, D4	**D2**	A2	D1 čeka, D2 kreće
1	D1, D3, D4	**D2**	-	D2 postavlja S
0	D3, D4	**D1**, **D2**	C1	D1 nastavlja, prioritetniji je
1	D3, D4	**D2**	-	D1 postavlja S i završava
0	D4	**D2**, D3	B2	D2 nastavlja
0	D2, D4	**D3**	A3	D2 čeka, D3 počinje
1	D2, D4	**D3**	-	D3 postavlja semafor
0	D4	**D2**, D3	C2	D2 nastavlja
1	D4	**D3**	-	D2 postavlja S i završava
0	-	**D3**, D4	B3	D3 nastavlja
0	D3	**D4**	-	D3 čeka S
0	D3	**D4**	A4	D4 počinje

S.v	Red Semafora	Red Pripravnih	Ispis	Komentar
1	D3	**D4**	-	D4 postavlja S
0	-	**D3**, D4	C3	D3 nastavlja
1	-	**D4**	-	D3 postavlja S i završava
1	-	**D4**	B4	D4 nastavlja
0	-	**D4**	-	D4 čeka S, ali S.v je 1, pa ga smanjuje i nastavlja
0	-	**D4**	C4	-
1	-	**-**	-	D4 postavlja S i završava

Dakle, prvih 5 ispisa je **A1, B1, A2, C1, B2**

b) Sadržaj i redoslijed reda pripravnih dretvi prilikom 10. ispisa:

Rj: Kao što vidimo, 10. ispis je C3. Red pripravnih dretvi je (po redu) **D3, D4.**

6. Faktor iskorištenja može poprimiti vrijednost ___ samo u potpuno determinističkom okruženju.

Rj: Odgovor je **1**. U nedeterminističkom okruženju to ne smije vrijediti jer je tamo ρ **prosječno** iskorištenje procesora, što znači da u nekom trenutku lagano možemo "zagušiti" procesor i prepuniti red čekanja.

7. Uz zadano prosječno zadržavanje posla u sustavu i prosječni broj dolazaka poslova u jedinici vremena možemo izračunati:

Rj: Prosječno zadržavanje posla je T, a prosječni broj dolazaka poslova u jedinici vremena je α. Stoga, prema Littleovom pravilu možemo dobiti **n**, jer **n = α*T**

Opaska:
Da je rečeno da je sustav Poisson-exp., mogli bismo dobiti i β preko $T = 1/(\beta - \alpha)$
Ipak, sustav nije definiran, a znamo da Littleovo pravilo vrijedi u općem slučaju.

8. Prosječni broj dolazaka u jedinici vremena za mješavinu više skupina poslova jednak je _____ prosječnih brojeva dolazaka pojedinačnih poslova.

Rj: Odgovor je, jasno **zbroju**, budući da vrijedi $\alpha = \alpha 1 + \ldots + \alpha n$

9. Za sinkronizaciju jednog proizvođača i više potrošača koji razmjenuju poruke preko *ograničenog međuspremnika* potrebno je:

Rj: Budući da je međuspremnik ograničen, koristi se **ciklički** što znači da su nam potrebna 2 OSEMa, jedan za brojanje slobodnih, a jedan za brojanje zauzetih mjesta. Da je on **neograničen**, ti OSEMovi nam ne bi bili potrebni. Drugi ključni izraz je "više potrošača". Budući da ih ima više, moramo dodati jedan BSEM koji će im služiti kao mutex, tj. osigurati da samo jedan troši poruke u isto vrijeme. Da imamo više proizvođača također bismo morali dodati jedan BSEM.

Konačan odgovor je, dakle, **2 OSEMa i 1 BSEM**

10. Posjetitelji ulaze u restoran u kojem ima R mjesta. U nekom trenutku u restoranu može biti najviše R posjetitelja, a ostali čekaju dok netko ne izađe. Kada iz restorana izađe N posjetitelja za redom (N > R) restoran se privremeno zatvara (novi posjetitelji ne mogu ući) jer je stolove potrebno očistiti. Za to je zadužena čistačica, koja tada čeka da svi posjetitelji izađu te nakon toga čisti stolove. Nakon čišćenja, restoran se ponovno otvara, za slijedećih N posjetitelja. Simulirajte sustav dretvama Posjetitelj(), koje se stvaraju nasumično, te jednom dretvom Čistačica(). Potrebno je napisati zadane monitorske strukture, te definirati podatkovnu strukturu i početne vrijednosti. Za ostvarenje monitora koristit funkcije *lock(M)*, *unlock(M)*, *wait(M, <uvjet>), signal(M, <uvjet>)* i *broadcast(M, <uvjet>)*. (funkcije imaju jednako djelovanje kao odgovarajuće UNIX funkcije).

Također zadano:

```
Posjetitelj():                  Čistačica();
m_fja Udji();                   ponavljaj {
jedi;                                   m_fja Cekaj();
m_fja Izadji();                         ocisti;
                                        m_fja Otvori();
                                }
```

Rj:
Varijable:

```
zatvoreno = false;
br_unutra = 0;
br_izaslo = 0;
mon. semafor M, red cekanja ulazak, red cekanja ciscenje;
```

Funkcije:

```
m_fja Udji() {
        lock(M);
        while(br_unutra >= R || zatvoreno == true) //Cekaj ako je puno ili zatvoreno
                wait(M, ulazak);
         br_unutra++; //udi
        unlock(M);
}
```

```
m_fja Izadji() {
        lock(M);
        br_unutra--; //izadi
        br_izaslo++;

        if(br_izaslo == N) signal(M, ciscenje); //Ako je N izaslo, javlja da pocisti
        signal(M, ulazak); //Javlja da je jedan izasao i da jedan smije uci

        /* Ako cistacica zatvara i ceka da svi izadu, javlja da jesu */
        if(zatvoreno == true && br_unutra == 0) signal(M, ciscenje);
        unlock(M);
}

m_fja Cekaj() {
        lock(M);
        wait(M, ciscenje); //Ceka da N-ti dojavi izlazak
        zatvoreno = true; //Zatvara
        wait(M, ciscenje); //Cekaj da svi izadu
        unlock(M);
}

m_fja Otvori() {
        lock(M);
        zatvoreno = false; //Otvori
        broadcast(M, ulazak); //Javi svima koji cekaju da je otvoreno
        unlock(M);
}
```

11. Disk s pokretnim glavama ima 100 staza (0-99). Neka se glava trenutno nalazi na 35. stazi, s time da je prethodno bila na stazi 15. Zahtjevi za pristup pojedinim stazama svrstani po redu prispjeća su 5, 22, 50, 14, 71, 44, 31, 90, 10, 82. Prilikom posluživanja zahtjeva disk uzima u obzir **samo prva četiri neposlužena zahtjeva u redu** (najstarija četiri neposlužena zahtjeva). Napisati redoslijed posluživanja zahtjeva po strategijama (SSTF, CLOOK, FCFS):

Rj:
Ovo boldano znači da u nekom trenutku u redu čekanja na posluživanje imamo **maksimalno** 4 zahtjeva, a kad se red isprazni za jedan zahtjev dolazi idući, po redu dolaska, dok ih ne ponestane.

SSTF je skraćenica od **S**hortest **S**eek **T**ime **F**irst. To znači da će idući posluženi zahtjev biti najbliži trenutnom. Smjer gibanja glave uzima se u obzir samo ako su dva zahtjeva jednako udaljena. Pogledajmo tablicu:

Pozicija	Red čekanja	Najbliži
35	5, 22, 50, 14	22
22	5, 50, 14, 71	14
14	5, 50, 71, 44	5
5	50, 71, 44, 31	31
31	50, 71, 44, 90	44
44	50, 71, 90, 10	50
50	71, 90, 10, 82	71
71	90, 10, 82	82
82	90, 10	90
90	10	10
10	-	-

CLOOK je **C**ircular **LOOK**. To znači da se, uzevši u obzir početni smjer gibanja glave (u ovom slučaju 15 → 35 – u desno) gibamo u desno dok god ima zahtjeva, zatim se vraćamo do početka (bez čitanja staza, **cirkularno**), te počevši od početka opet čitamo u desno. Pogledajmo tablicu:

Pozicija	Red čekanja
35	5, 22, 50, 14
50	5, 22, 14, 71
71	5, 22, 14, 44
5	22, 14, 44, 31
14	22, 44, 31, 90
22	44, 31, 90, 10
31	44, 90, 10, 82
44	90, 10, 82

82	90, 10
90	10
10	-

Plavom bojom označeni su retci gdje je CLOOK došao do zadnjeg zahtjeva na desnoj strani i "cirkularno" se vratio tražiti od početka.

FSCS je skraćenica od **F**irst **C**ome **F**irst **S**erved, što je isto kao i FIFO. Dakle, ovaj zadatak je trivijalan – zadatci se poslužuju istim redom kojim dolaze, dakle odgovor je prepisati zadani redoslijed dolazaka: **35, 5, 22, 50, 14, 71, 44, 31, 90, 10, 82**

12. Napišite funkciju Ograda() koja će sinkronizirati skup od N dretvi tako da ni jedna ne izađe iz funkcije prije nego je sve dretve iz skupa pozovu. Funkciju napišite uz korištenje potrebnog broja semafore (BSEM, OSEM) i varijabli. Navedite sve potrebne podatke i početne vrijednosti.

Rj:
Postoji više načina da se ovo riješi. Ja ću navesti dva – sa i bez OSEM-a. U oba slučaja nam je potreban BSEM jer **moramo štititi kritičnu varijablu brojača**. Dobar rule of thumb je **uvijek čitanje i pisanje kritične varijable vršiti pod mutexom, odnosno BSEMom**.

a) Bez OSEM

```
brojac = 0;
BSEM mutex, turnstile;
mutex.v = 1, turnstile.v = 0;

Ograda() {
	mutex.wait();
		brojac++;
		if(brojac == N)
			turnstile.signal();
	mutex.signal();

	turnstile.wait();
	turnstile.signal(); //Jednom kad prvi uspije proci, signalizira svima ostalima da smiju
}
```

b) Sa OSEM

```
brojac = 0;
BSEM mutex; (v = 1);
OSEM ograda; (v = 0);

Ograda() {
        mutex.wait();
        brojac++;
        if(brojac == N)
                for(i  = 1 : N)
                        ograda.signal(); //Povecaj OSEM za N;
        mutex.signal();

        ograda.wait();
}
```