Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (SI3OS2)
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo
Kolokvijum: Prvi, oktobar 2012.
Datum: 28.10.2012.

Prvi kolokvijum iz Operativnih sistema 2

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 2 sata. Dozvoljeno je korišćenje literature.

Zadatak 1 _____/10 *Zadatak 3* _____/10
Zadatak 2 _____/10

Ukupno: _____/30 = _____%

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po sistemu "sve ili ništa", odnosno nema parcijalnih poena. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**.

1. (10 poena) Raspoređivanje procesa

U nekom sistemu koristi se *Multilevel Feedback-Queue Scheduling* (MFQS), uz dodatni mehanizam koji ga približava ponašanju SJF algoritma. Svakom procesu pridružena je procena trajanja narednog naleta izvršavanja τ koja se izračunava eksponencijalnim usrednjavanjem sa α=1/2, uz odsecanje na ceo broj.

- Postoje tri reda spremnih procesa: HP (*High Priority*), MP (*Medium Priority*) i LP (*Low Priority*).
- Globalni algoritam raspoređivanja je po prioritetu, s tim da HP ima najviši, a LP najniži prioritet.
- Raspoređivanje po redovima je sledeće: za HP je *Round-Robin* (RR) sa vremenskim kvantumom 5, za MP je RR sa vremenskim kvantumom 10, a za LP je FCFS.
- Novoaktivirani proces (deblokiran ili kreiran) smešta se u red prema proceni τ: ako je $\tau \leq 5$, smešta se u HP, ako je $5 < \tau \leq 10$, smešta se u MP, inače se smešta u LP.
- Proces kome je istekao vremenski kvantum premešta se u red nižeg prioriteta od onog iz koga je uzet na izvršavanje. Procena τ se ažurira kada se završi ceo nalet izvršavanja, odnosno kada se proces blokira.

Posmatra se novoaktivirani proces P sa inicijalnom procenom τ=12 koji ima sledeću karaktekteristiku narednog izvršavanja (R*n* označava jedan nalet izvršavanja u trajanju *n* jedinica vremena, B označava I/O operaciju):
R2, B, R7, B, R2, B, R7, B, R7
Napisati sekvencu koja označava redove spremnih procesa u kojima redom boravi P, tako da za svako smeštanje procesa P u neki red u sekvenci postoji jedan element. Na primer, odgovor može da bude u obliku: HP, MP, LP, HP, ...

Odgovor: _____

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Jedna varijanta uslovne sinhronizacije unutar monitora je sledeća. Svaki monitor ima samo jednu, implicitno definisanu i anonimnu (bez imena) uslovnu promenljivu, tako da se u monitoru mogu pozivati sledeće dve sinhronizacione operacije:

- `wait()`: bezuslovno blokira pozivajući proces i oslobađa ulaz u monitor;
- `notifyAll()`: deblokira *sve* procese koji čekaju na uslovnoj promenljivoj, ako takvih ima (naravno, mešusobno isključenje tih procesa je i dalje obezbeđeno).

Projektuje se konkurentni klijent-server sistem. Server treba modelovati monitorom sa opisanom uslovnom promenljivom. Klijenti su procesi koji ciklično obavljaju svoje aktivnosti. Pre nego što u jednom ciklusu neki klijent započne svoju aktivnost, dužan je da od servera traži dozvolu u obliku "žetona" (*token*). Kada dobije žeton, klijent započinje aktivnost. Po završetku aktivnosti, klijent vraća žeton serveru. Server vodi računa da u jednom trenutku ne može biti izdato više od N žetona: ukoliko klijent traži žeton, a ne može da ga dobije jer je već izdato N žetona, klijent se blokira. Napisati kod monitora i procesa-klijenta.

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Implementirati veb servis, na programskom jeziku Java, koji će krajnjem korisniku pružiti sledeći interfejs:

```
public class NetBuffer {
 public NetBuffer(String host, int port)
 public void put(int d[]);
 public int get();
 public int getK();
}
```

Servis treba da obezbedi usluge ograničenog kružnog bafera u koji se može atomično umetati K celobrojnih podataka, gde je K konstanta koja je definisana pri kreiranju tog bafera. `NetBuffer` se povezuje sa odgovarajućim serverom preko koga se vrši razmena podataka. Metode `put` i `get` imaju semantiku primitiva za rad sa ograničenim kružnim baferom, a služe za atomično umetanje K elemenata, odnosno za dohvatanje jednog elementa iz bafera koji se nalazi na serveru. Metoda `getK` dohvata i postavlja konstantu K za koju je udaljeni bafer kreiran. Na raspolaganju je klasa `BoundedBuffer` koja implementira navedeni ograničeni kružni bafer i ima sledeći interfejs:

```
public class BoundedBuffer {
    public BoundedBuffer (int N, int K) ;
    public synchronized void put(int d[]);
    public synchronized int get();
}
```

Parametar N u konstruktoru predstavlja dimenziju bafera. Poznato je da je K<N, ali nije poznato da li je N deljivo sa K. Međuprocesnu komunikaciju realizovati preko priključnica (*socket*) i razmenom poruka (*message passing*). Dozvoljeno je korišćenje koda prikazanog na vežbama (kod sa vežbi ne treba prepisivati, nego precizno navesti koja klasa ili koji metod se koriste i/ili menjaju, nasleđuju, ...). Nije potrebno proveravati uspešnost izvršavanja operacija (*try/catch* klauzule).

Rešenje: