Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

*Predmet:* Operativni sistemi 2
*Nastavnik:* prof. dr Dragan Milićev
*Odsek:* Softversko inženjerstvo, Računarska tehnika i informatika
*Kolokvijum:* Treći, januar 2019.
*Datum:* 14. 1. 2019.

# *Treći kolokvijum iz Operativnih sistema 2*

*Kandidat:* _____

*Broj indeksa:* _____ *E-mail:*_____

*Kolokvijum traje 1,5 sat. 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) Upravljanje diskovima

Klasa `CLook`, čiji je interfejs dat dole, implementira algoritam *C-Look* raspoređivanja zahteva za operacije na disku. Svaki zahtev (struktura `Req`) odnosi se na dati cilindar (`Req::cyl`). Zahtevi su u redu zahteva unutar objekta klase `CLook` uvezani u dvostruko ulančanu cirkularnu listu (polja `Req::next` i `Req::prev`), radi lakšeg obilaska. Operacija `CLook::get` vraća zahtev koji treba opslužiti, ali ne uklanja taj zahtev iz liste, sve dok se zahtev ne opsluži do kraja; tek kada se završi obrada tekućeg zahteva, on će biti uklonjen iz liste pozivom operacije `CLook::remove`. Ovo je zbog toga što se tokom opsluživanja jednog zahteva u red mogu stavljati novi zahtevi (operacijom `CLook::put`) koji se odnose na isti cilindar ili na cilindre koji su nailazeći, kako bi se i oni opslužili u istom tekućem prolazu glave diska.

Implementirati operacije `get` i `remove`.

```
struct Req {
  Req *prev, *next;
  unsigned cyl;
};

class CLook {
public:
  CLook ();

  Req* get () const;
  void remove ();
  void put (Req*);
  ...
};
```

Rešenje:

## 2. (10 poena) Operativni sistem Linux

Fajl `/proc/cpuinfo` sadrži informacije o trenutnom stanju procesora. Za svako jezgro koje procesor poseduje u fajlu postoji deo koji opisuje stanje tog jezgra. Primer dela za jedno jezgro dat je u nastavku (isti format je i za ostala jezgra):

```
processor  : 0
vendor_id  : GenuineIntel
cpu family : 6
model      : 158
model name : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
stepping   : 10
microcode  : 0x96
core MHz         : 4299.982
cache size : 12288 KB
physical id      : 0
siblings   : 12
core id          : 0
```

Napisati *shell* skriptu koja izračunava trenutnu prosečnu frekvenciju svih jezgara. Računjanje vršiti na nivou celih brojeva sa odsecanjem.

Rešenje:

## 3. (10 poena) Operativni sistem Linux

Napisati program za operativni sistem Linux koji ubacuje N pseudoslučajnih brojeva u stablo. Stablo treba napraviti u fajlu čije je ime zadato kao prvi argument prilikom pokretanja programa (smatrati da fajl postoji i da je dovoljno velik da stablo može u njega da stane). Obezbediti da napravljeno stablo može koristiti neki drugi program u budućnosti. Veličina adrese u datom sistemu, kao i tipa `unsigned long`, je 64 bita.

Data je klasa `Tree` koja realizuje pravljenje stabla. Deo implementacije dat je u nastavku:

```
class Tree {
public:
    Tree(void *startAddress, long size) :
            segmentStartAddress(startAddress),
            segmentSize(size) {}
    virtual void *getLogicalAddress(void *psysicalAddress) = 0;
    virtual void *getVirtualAddress(void *logicalAddress) = 0;
    void addNumber(int n);
    ...

protected:
    void *segmentStartAddress;
    long segmentSize;
};
```

Data klasa kreira stablo u virtuelnom adresnom prostoru počevši od adrese koja je prosleđena konstruktoru. Veličina prostora za stablo se prosleđuje konstruktoru (za potrebe ovog zadatka dovoljna veličina je 10000). Metoda `addNumber` ubacuje jedan broj u stablo. U čvorovima stabla se ne čuva virtuelna adresa potomka, već adresa koju vraća metoda `getLogicalAddress` kojoj se prosledi virtuelna adresa potomka. Metoda `getVirtualAddress` se poziva prilikom obilaska stabla za potrebe određivanja virtuelne adrese potomka. Njoj se kao parametar prosleđuje adresa potomka koja je zapamćena u čvoru. Nije potrebno proveravati ispravnost poziva funkcija.

Rešenje: