Drugi kolokvijum iz Operativnih sistema 1
Septembar 2013.

1. (10 poena)

```
#include <semaphore.h>

void main () {
  sem_t* mutex = sem_open(”mutex”,1);
  if (mutex==0) return; // Exception!

  <noncritical_section>

  int status = sem_wait(mutex);
  if (status<0) ... // Exception!
  <critical_section>
  int status = sem_post(mutex);
  if (status<0) ... // Exception!

  <noncritical_section>
  sem_unlink(mutex);
}
```

2. (10 poena)

```
void Event::wait () {
  lock(lck);
  if (--val<0) {
    blocked.put(runningUserThread);
    runningUserThread = Scheduler::get();
  }
  unlock(lck);
}


void Event::signal () {
  lock(lck);
  if (val++<0)
    Scheduler::put(blocked.get());
  else val = 1;
  unlock(lck);
}
```

3. (10 poena)

```
class DLArray {
public:
  DLArray (int size, int blockSize, FHANDLE fromFile);

  inline double get (int i);  // Get element [i]
  inline void   set (int i, double x); // Set element [i]

protected:
  inline void save(int slot);
  inline void load(int blockNo, int slot);
  inline int  fetch(int blockNo);

private:
  FHANDLE file;
  int size, blockSize;
  int curBlock[2];
  int dirty[2];
  double* block[2];
```

```
};


DLArray::DLArray (int s, int bs, FHANDLE f) :
  file(f), size(s), blockSize(bs) {
  for (int slot=0; slot<2; slot++) {
    curBlock[slot]=slot;
    dirty[slot]=0;
    block[slot] = new double[bs];
    load(curBlock[slot],slot);
  }
}

void DLArray::save (int slot) {
  fwrite(file,curBlock[slot]*blockSize,block[slot],blockSize);
  dirty[slot]=0;
}


void DLArray::load (int b, int slot) {
  curBlock[slot] = b;
  fread(file,curBlock[slot]*blockSize,block[slot],blockSize);
  dirty[slot] = 0;
}


int DLArray::fetch(int b) {
  int slot = b%2;
  if (curBlock[slot]!=b) {
    if (dirty[slot]) save(slot);
    load(b,slot);
  }
  return slot;
}

double DLArray::get (int i) {
  if (block==0 || i<0 || i>=size) return 0; // Exception
  int slot = fetch(i/blockSize);
  return block[slot][i%blockSize];
}

void DLArray::set (int i, double x) {
  if (block==0 || i<0 || i>=size) return; // Exception
  int slot = fetch(i/blockSize);
  if (block[slot][i%blockSize]!=x) {
    block[slot][i%blockSize]=x;
    dirty[slot]=1;
  }
}
```

4. (10 poena)

a)(3) VA(64): Page1(25):Page2(25):Offset(14).
PA(40): Frame(26):Offset(14).

b)(3) Širina PMT2 je 32 bita, od koga 26 sadrže broj okvira, 2 bite zaštite, a ostali su neiskorišćeni. Ista je i širina PMT1.
PMT1 ima 2^{25} ulaza širine 32 bita (4B), što je ukupno: 2^{27}B=128MB.

c)(4) Ovaj proces koristio je 2^{38} svojih najnižih adresa, što je $2^{38-14}=2^{24}$ stranica. Jedna PMT drugog nivoa pokriva 2^{25} stranica, pa je ovaj proces alocirao PMT prvog nivoa i jednu PMT drugog nivoa. Zato ukupna veličina PMT iznosi: 2·128MB=256MB.