Rešenja zadataka za
drugi kolokvijum iz Operativnih sistema 1
April 2018.

1. (10 poena)

```
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>

class Mutex {
public:
  Mutex (const char *name);
 ~Mutex ();
  int entry ();
  int exit();
private:
  sem_t* mutex;
  pid_t pid;
};

Mutex::Mutex (const char *name) {
  this->mutex = sem_open(name,O_CREAT,1);
}

int Mutex::entry () {
  if (!this->mutex) return -1;
  int ret = sem_wait(this->mutex);
  if (ret<0) return ret;
  this->pid = getpid(); // Must not be done before sem_wait
  return 0;
}

int Mutex::exit () {
  if (!this->mutex || this->pid!=getpid()) return -1; // error
  return sem_post(this->mutex);
}

Mutex::~Mutex () {
  sem_close(this->mutex);
}
```

2. (10 poena)

```
int proc_relocate (PCB* pcb) {
  FreeMem *cur=fmem_head, *prev=0;
  for (; cur && (char*)cur+cur->size!=pcb->mem_base_addr;
         prev=cur, cur=cur->next);

  if (cur==0 || cur->next==0 ||
      (char*)cur->next!=pcb->mem_base_addr+pcb->size)
    // No room or no need for relocation
    return 0;

  // Relocate the process to cur
  // and join the two fragments (cur and cur->next):

  size_t new_size = cur->size + cur->next->size;
  FreeMem* new_next = cur->next->next;

  memcpy(cur,pcb->mem_base_addr,pcb->mem_size);
  pcb->mem_base_addr = cur;

  if (prev) {
    prev->next = (FreeMem*)(pcb->mem_base_addr+pcb->mem_size);
    prev->next->next = new_next;
    prev->next->size = new_size;
  } else {
    fmem_head = (FreeMem*)(pcb->mem_base_addr+pcb->mem_size);
    fmem_head->next = new_next;
    fmem_head->size = new_size;
  }

  return 1;
}
```

3. (10 poena)

VA(64): Page1(25):Page2(25):Offset(14)

PA(40): Frame(26):Offset(14)

```
const unsigned short pg1w = 25, pg2w = 25, offsw = 14;

inline unsigned getPMT1EntryForPage (unsigned long pg) {
  return pg>>pg2w;
}

inline unsigned getPMT2EntryForPage (unsigned long pg) {
  return pg & ~(-1L<<pg2w);
}

void setPMTEntryForPage (unsigned* pmt1, unsigned long pg, unsigned val) {
  unsigned pmt1entry = getPMT1EntryForPage(pg);
  if (pmt1[pmt1entry]==0)
    pmt1[pmt1entry] = alloc_pmt2();
  unsigned* pmt2 = (unsigned*)(((unsigned long)pmt1[pmt1entry])<<offsw);
  pmt2 = (unsigned*)kaddrPtoV(pmt2);
  unsigned pmt2entry = getPMT2EntryForPage(pg);
  pmt2[pmt2entry] = val;
}

void initPMT (unsigned* pmt, unsigned long pageFrom, unsigned long nPages,
              unsigned long frameFrom, unsigned short rwx) {
  for (unsigned long i=0; i<nPages; i++) {
    unsigned descr = (frameFrom+i) | (((01<<3)|rwx)<<27);
    setPMTEntryForPage(pmt,pageFrom+i,descr);
  }
```