Rešenja zadataka za drugi kolokvijum iz Operativnih sistema 1 Septembar 2015.

1. (10 poena)

```
void Semaphore::wait (unsigned int n = 1) {
  lock(lck);
  val -= n;
  if (val<0) {
    Thread::running->waiting = -val;
    val = 0;
    block();
  }
  unlock(lck);
}

void Semaphore::signal () {
  lock(lck);
  Thread* thr = blocked.first();
  if (thr) {  // There are blocked threads, val==0
    thr->waiting--;
    if (thr->waiting==0)
      deblock();
  } else // No blocked threads, val>=0
    val++;
  unlock(lck);
}
```

2. (10 poena)

Jednostavnije, ali manje efikasno rešenje:

```
// Helper: Try to join cur with the cur->next free segment:
int tryToJoin (FreeMem* cur) {
  if (!cur) return 0;
  if (cur->next && (char*)cur+cur->size == (char*)(cur->next)) {
    // Remove the cur->next segment:
    cur->size += cur->next->size;
    cur->next = cur->next->next;
    if (cur->next) cur->next->prev = cur;
    return 1;
  } else
    return 0;
}
```

```
int mem_free(char* addr, size_t size) {
  // Find the place where to insert the new free segment (just after cur):
  FreeMem* cur=0;
  if (!fmem_head || addr<(char*)fmem_head)
    cur = 0; // insert as the first
  else
    for (cur=fmem_head; cur->next!=0 && addr>(char*)(cur->next);
         cur=cur->next);

  // Insert the new segment after cur:
  FreeMem* newSeg = (FreeMem*)addr;
  newSeg->size = size;
  newSeg->prev = cur;
  if (cur) newSeg->next = cur->next;
  else newSeg->next = fmem_head;
  if (newSeg->next) newSeg->next->prev = newSeg;
  if (cur) cur->next = newSeg;
  else fmem_head = newSeg;

  // Try to merge with the previous and next segments:
  tryToJoin(newSeg);
  tryToJoin(cur);
}
```

Složenije, ali nešto efikasnije rešenje:

```
int mem_free(char* addr, size_t size) {
  // Find the place where to insert the new free segment (just after cur):
  FreeMem* cur=0;
  if (!fmem_head || addr<(char*)fmem_head)
    cur = 0; // insert as the first
  else
    for (cur=fmem_head; cur->next!=0 && addr>(char*)(cur->next);
         cur=cur->next);

  // Try to append it to the previous free segment cur:
  if (cur && (char*)cur+cur->size==addr) {
    cur->size+=size;
    // Try to join cur with the next free segment:
    if (cur->next && (char*)cur+cur->size == (char*)(cur->next)) {
      // Remove the cur->next segment:
      cur->size += cur->next->size;
      cur->next = cur->next->next;
      if (cur->next) cur->next->prev = cur;
    }
    return;
  }
  else

  // Try to append it to the next free segment:
  FreeMem* nxtSeg = cur?cur->next:fmem_head;
  if (nxtSeg && addr+size==(char*)nxtSeg) {
    FreeMem* newSeg = (FreeMem*)addr;
    newSeg->size = nxtSeg->size+size;
    newSeg->prev=nxtSeg->prev;
    newSeg->next=nxtSeg->next;
    if (nxtSeg->next) nxtSeg->next->prev = newSeg;
    if (nxtSeg->prev) nxtSeg->prev->next = newSeg;
    else fmem_head = newSeg;
    return;
  }
  else

  // No need to join; insert the new segment after cur:
  FreeMem* newSeg = (FreeMem*)addr;
  newSeg->size = size;
  newSeg->prev = cur;
  if (cur) newSeg->next = cur->next;
  else newSeg->next = fmem_head;
  if (newSeg->next) newSeg->next->prev = newSeg;
  if (cur) cur->next = newSeg;
  else fmem_head = newSeg;
}
```

3. (10 poena)

a)(3)	VA(60):	Page1(16):Page2(16):Page3(16):Offset(12).
	PA(42):	Frame(30):Offset(12).

b)(7)

```
const unsigned short pg1w = 16, pg2w = 16, pg3w = 16, offsw = 12;
const unsigned pmt1Size = 1U<<pg1w,
               pmt2Size = 1U<<pg2w, pmt3Size = 1U<<pg3w;
typedef unsigned long ulong;

void deallocateAllFrames (PCB* pcb) {
  if (pcb==0) return;  // Exception

  // Traverse PMT1:
  unsigned* pmt1 = (unsigned*)((ulong)(pcb->pmt1)<<offsw);
  for (unsigned i1 = 0; i1<pmt1Size; i1++) {
    if (pmt1[i1]==0) continue;

    // Traverse PMT2:
    unsigned* pmt2 = (unsigned*)((ulong)(pmt1[i1])<<offsw);
    for (unsigned i2 = 0; i2<pmt2Size; i2++) {
      if (pmt2[i2]==0) continue;

      // Traverse PMT3:
      unsigned* pmt3 = (unsigned*)((ulong)(pmt2[i2])<<offsw);
      for (unsigned i3 = 0; i3<pmt3Size; i3++) {
        if (pmt3[i3]==0) continue;
        // Deallocate frame:
        unsigned frame = pmt3[i3]>>2;
        deallocateFrame(frame);
      }
    }
  }
}
```