Rešenja drugog kolokvijuma iz Operativnih sistema 2, septembar 2014.

1. (10 poena)

```
int ResourceAllocator::announce (int p, int r) {
  if (p<0 || p>=np || r<0 || r>=nr) return -1; // Exception
  if (alloc[p][r]!=0) return 0; // No effect if already used
  alloc[p][r] = 1;  // Announcement edge
  return 1;
}

int ResourceAllocator::request (int p, int r) {
  if (p<0 || p>=np || r<0 || r>=nr) return -1; // Exception
  if (alloc[p][r]==0) return -2; // Error: Request without announcement
  if (alloc[p][r]==-1) return -3; // Error: Resource already acquired
  alloc[p][r] = 2;  // Request edge
  for (int j=0; j<np; j++)
    if (alloc[j][r]==-1) return 0;  // Resource already occupied
  alloc[p][r] = -1; // Try to allocate it and check for a cycle
  if (hasCycle()) {
    alloc[p][r] = 2;
    return 0;  // Cannot be acquired due to a possible deadlock
  } else
    return 1;  // Resource acquired
}

inline int ResourceAllocator::release (int p, int r) {
  if (p<0 || p>=np || r<0 || r>=nr) return -1; // Exception
  if (alloc[p][r]!=-1) return -2; // Resource not occupied
  alloc[p][r] = 1;  // Announcement edge
  return 1;
}
```

2. (10 poena)

```
void updateReferenceRegs (PCB* pcb) {
  if (pcb==0) return; // Exception!
  for (unsigned page=0; page<PMTSIZE; page++) {
    unsigned frame = pcb->pmt[page];
    unsigned ref = pcb->pageRefHash.getValue(page);
    ref >>= 1;
    if (frame&1) ref |= ~MAXINT;
    pcb->pageRefHash.setValue(page,ref);
    (pcb->pmt[page]) &= ~1U;  // Reset the reference bit
  }
}
```

3. (10 poena)

```
template<class T, int slabSize>
void* SlabAllocator<T,slabSize>::alloc () {
  for (Slab* s = cacheHead; s!=0; s=s->next)
    for (int i=0; i<numOfSlots; i++)
      if (s->slots[i].isFree) {
        // Found a slot, return it:
        s->slots[i].isFree = 0;
        return &(s->slots[i].slot);
      }
  // Not found, allocate a new slab:
  Slab* s = new Slab();
  if (s==0) return 0; // Out of memory exception!
  s->next = cacheHead;
  s->prev = 0;
  if (cacheHead==0)
    cacheTail = s;
  else
    cacheHead->prev = s;
  cacheHead = s;
  // And return the first slot:
  s->slots[0].isFree = 0;
  return &(s->slots[0].slot);
}
```