Rešenja drugog kolokvijuma iz Operativnih sistema 2, decembar 2018.

1. (10 poena)

```
struct RCB {
  unsigned id;
  int free; // Is this resource free (available for allocation)?
  RCB* next;
  RCB* prev;
  ...
};

struct PCB {
  RCB* allocr; // List of allocated resources, sorted by id
  ...
};

int resource_allocate (PCB* p, RCB* r) {
  if (!p || !r) return -2; // Exception;
  if (r->free) { // Resource free, allow allocation and record it
    r->free = 0;
    // Put r into the process's sorted resource allocation list
    RCB* cur = p->allocr;
    if (!cur || r->id >= cur->id) {
      r->next = cur;
      r->prev = 0;
      if (cur) cur->prev = r;
      p->allocr = r;
    } else {
      while (cur->next && r->id < cur->next->id) cur = cur->next;
      r->next = cur->next;
      r->prev = cur;
      if (cur->next) cur->next->prev = r;
      cur->next = r;
    }
    return 1;
  }
  // The resource is already allocated:
  if (!p->allocr || r->id > p->allocr->id) return 0; // Allowed to wait
  else return -1; // Not allowed to wait, to avoid deadlock
}

void resource_free (PCB* p, RCB* r) {
  if (!p || !r) return -2; // Exception;
  r->free = 1;
  if (r->prev) r->prev->next = r->next;
  else p->allocr = r->next;
  if (r->next) r->next->prev = r->prev;
  r->next = r->prev = 0;
}
```

2. (10 poena)

```
PageDesc* getFromGroup(int group, unsigned long start) {
  for (i = 0; i<NumOfPages; i++) {
      PageDesc* pd = &pages[(start+i)%NumOfPages];
      if (pd->flags&3 == group)
        return pd;
  }
  return 0;
}
PageDesc* getVictim () {
  unsigned long start = rand()%NumOfPages;
  int group;
  for (group = 0; group < 4; group ++) {
    PageDesc* pd = getFromGroup(group, start);
    if (pd != 0)
      return pd;
  }
  return 0; // Exception
}
```

3. (10 poena)

```
Slot* Cache::alloc () {
  Slab* cur = 0;
  for (cur = this->head; cur; cur = cur->next)
    if (cur->numOfFreeSlots>0 && cur->numOfFreeSlots<SlabSize) break;
  if (!cur)
    for (cur = this->head; cur; cur = cur->next)
      if (cur->numOfFreeSlots>0) break;
  if (!cur)
    cur =  this->head = new Slab(this->head); // May throw an exception
  Slot* slot = &cur->slots[cur->head];
  cur->numOfFreeSlots--;
  cur->head = *(long*)slot;
  return slot;
}
};
```