Rešenja zadataka za kolokvijum iz Operativnih sistema 1 avgust 2020.

1. **(10 poena)** Pokazana je jedna jednostavna implementacija koja zadovoljava uslove, koristi jedostavan FIFO red zahteva, ali koja ima značajan stepen odbijanja zahteva (red zahteva je opadajući po vremenskoj marki). Moguće su i složenije varijante sa boljim učinkom (manje odbijenih zahteva), ali treba paziti na moguće izgladnjivanje.

```
typedef unsigned long Time;
const Time MAXTIME = (Time)-1;

class Resource {
public:
  Resource () : holder(0), last(MAXTIME) {}
  int  acquire();
  void release();
protected:
  void block (); // Implementacija ista kao i za Semaphore
  void deblock (); // Implementacija ista kao i za Semaphore
private:
  Thread* holder;
  Time last;
  ThreadQueue blocked;
};

void Resource::acquire () {
  lock();
  if (!holder) {
    holder = Thread::running;
    last = holder->timestamp;
    unlock();
    return 1;
  } else
  if (Thread::running->timestamp>=last) {
    unlock();
    return 0;
  } else {
    last = Thread::running->timestamp;
    block();
    unlock();
    return 1;
  }
}

void Resource::release () {
  lock();
  Thread* thr = blocked.first();
  if (thr) {
    holder = thr;
    deblock();
  } else {
    holder = 0; last = MAXTIME;
  }
  unlock();
}
```

2. (10 poena)

a)(5)

```
SegDesc* findSegDesc (SegDesc* root, size_t page) {
  SegDesc* sd = root;
  while (!sd) {
    if (sd->pg<=page && page<sd->pg+sd->sz) return sd;
    if (page<sd->pg) sd = sd->left;
    else sd = sd->right;
  }
  return nullptr;
}
```

b)(5)

```
int handlePageFault (PCB* pcb, size_t page) {
  SegDesc* sd = findSegDesc(pcb->segdesc,page);
  if (!sd) return -1; // Memory access violation
  size_t frame = allocateFrame();
  if (!frame) return -2; // No free memory
  int s = sd->vtp->loadPage(page,frame);
  if (s<0) return s; // Error loading page
  setPMTEntry(pcb,page,frame,sd);
  return 0;
}
```

3.

```
static REG* ptr = 0;  // pointer to current data item
static int count = 0; // counter

void startIO () { // Helper: start a new transfer
  if (ioHead==0) return;
  ptr = ioHead->buffer;
  count = BlockSize;
  *ioCtrl = 1; // Start I/O
}

void transfer () {
  startIO();
}

interrupt void ioInterrupt () {
  if (*ioStatus&2)
    ioHead->status = -1;  // Error in I/O
  else {  // Transfer the next data item
    *ptr++ = *ioData;
    if (--count)
      return;
    else
      ioHead->status = 0;  // Transfer completed successfully
  }
  *ioCtrl = 0; // Stop the transfer
  ioHead = ioHead->next; // Remove the request from the list
  startIO();  // Start a new transfer
}
```

4. (10 poena)

```
int write (int fd, Node* root) {
  if (!root) return 0;
  int ret = write(root->left);
    if (ret<0) return ret;
```

```
  ret = write(fd,&(root->contents),sizeof(int));
    if (ret<0) return ret;
  ret = write(root->right);
    return ret;
}

int save (const char* fname, Node* root) {
  int fd = open(fname,O_WRONLY|O_CREAT|O_TRUNC,
                      S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
  if (fd<0) return fd;
  int ret = write(fd,root);
  close(fd);
  return ret;
}
```