Rešenja prvog kolokvijuma iz Operativnih sistema 2 decembar 2023.

1. (10 poena) U klasu `PCB` treba dodati podatke članove `age` (broj ranijih boravaka procesa u istom redu) i `lastPri` (indeks reda u kom je proces bio prethodni put), oba tipa `int`, inicijalizovani na 0 (nije neophodno za dato rešenje, pod uslovom da se proces stavlja prvi put u red spremnih sa `wasBlocked=true`).

```
class Scheduler {
public:
  Scheduler () {}
  PCB* get ();
  void put (PCB*, bool wasBlocked);

private:
  class ProcList {
  public:
    ProcList () : head(0), tail(0) {}
    void put (PCB* p);
    PCB* get ();
  private:
    PCB *head, *tail;
  };

  static const int HP, MP, LP;
  ProcList ready[3];
};

const int Scheduler::HP = 0, Scheduler::MP = 1, Scheduler::LP = 2;

inline void Scheduler::ProcList::put (PCB* p) {
  if (tail) tail = tail->next = p;
  else head = tail = p;
  p->next = 0;
}

inline PCB* Scheduler::ProcList::get () {
  PCB* ret = head;
  if (head) head = head->next;
  if (!head) tail = 0;
  return ret;
}

PCB* Scheduler:: put (PCB* p, bool wasBlocked) {
  int pri = HP;
  if (wasBlocked)
    p->age = 0;
  else {
    pri = p->lastPri;
    if (++p->age > AGE_THR) {
      p->age = 0;
      if (pri<LP) pri++;
    }
  }
  p->lastPri = pri;
  ready[pri].put(p);
}
```

```
PCB* Scheduler::get () {
  for (int i=HP; i<=LP; i++) {
    PCB* p = ready[i].get();
    if (p) return p;
  }
  return 0;
}
```

2. (10 poena)

```
monitor ReadersWriters;
  export startRead, stopRead, startWrite, stopWrite;

  var writers : condition;
      waitingWriters : integer;
      isWriting : boolean;
      readers : condition;
      readersCount : integer;

  procedure startRead;
  begin
    if isWriting or waitingWriters>0 then
      readers.wait;
    readCount := readCount + 1;
  end;

  procedure stopRead;
  begin
    readCount := readCount - 1;
    if readCount=0 then
      if waitingWriters>0 then
        writers.signal;
      else
        readers.signalAll;
  end;

  procedure startWrite;
  begin
    if isWriting or readersCount>0 then
    begin
      waitingWriters := waitingWriters + 1;
      writers.wait;
      waitingWriters := waitingWriters - 1;
    end;
    isWriting := true;
  end;

  procedure stopWrite;
  begin
    isWriting := false;
    if waitingWriters>0 then
      writers.signal;
    else
      readers.signalAll;
  end;

begin
  isWriting := false; readersCount := 0; waitingWriters := 0;
end;
```

3. (10 poena)

```
public class Server {
    public Map<String, Integer> userLogged = new HashMap<>();

    public void run() {
        ServerSocket serverSocket = null;

        try {
            serverSocket = new ServerSocket(5555);

            while (true) {
                Socket clientSocket = serverSocket.accept();

                Service service = new Service(clientSocket);
                Thread t = new RequestHandler(this, service);
                t.start();

            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public synchronized void login(String user) {
        if (!userLogged.containsKey(user)) {
            userLogged.put(user, 0);
        }


        int logged = userLogged.get(user);
        while (logged >= 3) {
            try {
                wait();
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            logged = userLogged.get(user);
        }

        userLogged.put(user, logged + 1);
    }

    public synchronized void signOut(String user) {
        if (userLogged.containsKey(user)) {
            int logged = userLogged.get(user);
            userLogged.put(user, Math.max(logged - 1, 0));
            notifyAll();
        }
    }

    public static void main(String[] args) {
        new Server().run();
    }
}

public class RequestHandler extends Thread {
    private final Server server;
    private final Service service;
    private String username = null;
```

```
    private boolean signOut = false;

    public RequestHandler(Server server, Service service) {
        this.server = server;
        this.service = service;
    }

    public void run() {
        try {
            login();

            // Use server...

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (signOut) {
                server.signOut(username);
            }
        }
        service.close();
    }

    private void login() throws IOException {
        while (true) {
            String msg = service.receiveMsg();
            if (msg == null) {
                throw new IOException("Msg null received");
            }
            String[] data = msg.split("#");

            if ("login".equals(data[1])) {
                username = data[2];
                server.login(username);
                signOut = true;
                break;
            }
        }
    }
}
```