Rešenja prvog kolokvijuma iz Operativnih sistema 2 decembar 2025.

1. (10 poena)

```
PCB* Scheduler::updateTime (unsigned pri, Time execTime) {
  times[pri] = (times[pri]>=execTime)?(times[pri]-execTime):0;
}

PCB* Scheduler::get () {
  for (unsigned i=0; i<MAXPRI; i++) {
    PCB* p = queues[i].get();
    if (!p) continue;
    Time slice = min(p->getDefaultTimeSlice(),times[i]);
    p->setTimeSlice(slice);
    return p;
  }
}
```

2. (10 poena)

```
class ReaderWriters {
public:
  ReaderWriters () {}
  const T* read () const;
  void write (const T&);
  void swap ();

private:
  T data[2];
  int toRead = -1, toWrite = 0;
  Semaphore mutex = 1;
};

const T* ReaderWriters::read () const {
  mutex.wait();
  T* ret = 0;
  if (toRead>=0) ret = &data[toRead];
  mutex.signal();
  return ret;
}

void ReaderWriters::write (const T& t) {
  mutex.wait();
  data[toWrite] = t;
  mutex.signal();
}

void ReaderWriters::swap () {
  mutex.wait();
  toWrite = 1 - toWrite;
  toRead = 1 - toWrite;
  mutex.signal();
}
```

3. (10 poena)

```
public class Server {
    public void run() {
        ServerSocket serverSocket = null;

        try {
            serverSocket = new ServerSocket(5555);

            while (true) {
                System.out.println("Wait for client");
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client accepted");

                Service service = new Service(clientSocket);

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

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

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

    public List<String> execute(String deviceId, List<String> commands) {
        int pos = deviceIds.indexOf(deviceId);
        if (pos >= 0) {
            String dev = deviceIds.get(pos);
            synchronized (dev) {
                return executeOnDevice(deviceId, commands);
            }
        } else {
            return null;
        }

    }

    private final List<String> deviceIds = new ArrayList<>();

    public List<String> executeOnDevice(String deviceId, List<String>
commands);
}

public class RequestHandler extends Thread {
    private final Service service;
    private final Server server;

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

    public void run() {
        while (true) {
            try {
                String msg = service.receiveMsg();
```

```
                if (msg == null) {
                    break;
                }
                String[] parts = msg.split("#");
                String deviceId = parts[0];
                int num = Integer.parseInt(parts[1]);
                List<String> commands = new ArrayList<>();
                boolean success = true;
                for (int i = 0; i < num; i++) {
                    String command = service.receiveMsg();
                    if (command == null) {
                        success = false;
                        break;
                    }
                    commands.add(command);
                }

                if (!success) {
                    break;
                }

                parseAndRespond(deviceId, commands);
            } catch (IOException e) {
                e.printStackTrace();
                break;
            }

        }

        service.close();
    }

    private void parseAndRespond(String deviceId, List<String> commands) {
        List<String> response = server.execute(deviceId, commands);
        if (response == null) {
            service.sendMsg("ERROR#Device not found");
        } else {
            service.sendMsg("OK#" + response.size());
            for (String line : response) {
                service.sendMsg(line);
            }
        }

    }
}
```

Klasa Service je ista kao na vežbama.