Rešenja prvog kolokvijuma iz Operativnih sistema 2 Januar 2019.

1. (10 poena)

```
class Scheduler {
public:
  Scheduler () {...}
  PCB* get ();

  void remove (PCB*);
  void put (PCB*);

private:
  ProcGroup *head, *tail;
  ProcGroup *runningGroup;
  PCB *runningProc;
};

PCB* Scheduler::get () {
  if (runningProc==0 || runningGroup==0) return 0; // Exception!
  runningProc = runningProc->next;
  while (!runningProc) {
    runningGroup = runningGroup->next;
    if (!runningGroup) runningGroup = head;
    runningProc = runningGroup->head;
  }
  runningProc->slice = runningGroup->slice/runningGroup->size;
  return runningProc;
}
```

2. (10 poena)

```
class TickTuck {
  private boolean canTuck = false;

  synchronized void tick () {
    // do tick
    canTuck = true;
    notify();
  }

  synchronized void tuck () {
    while (!canTuck) wait();
    // do tuck
    canTuck = false;
  }
}
```

3. (10 poena)

```
public class Server extends Thread{
    private final Service input;
    private final Map<String, String> m_messages = new HashMap<String,
String>();

    public Server(Socket input) throws IOException {
        this.input = new Service(input);

        start();
        work();
    }
```

```
    private void work() throws IOException {
        ServerSocket server = new ServerSocket(5555);
        while (true) {
            Socket client = server.accept();
            Service service = new Service(client);
            Thread reqHandler = new RequestHandler(this, service);
            reqHandler.start();
        }
    }

    public void run() {
        while (true) {
            String key = input.readMessage();
            String msg = input.readMessage();

            addMessage(key, msg);
        }
    }

    public synchronized void addMessage(String key, String msg) {
        m_messages.put(key, msg);
        notifyAll();
    }

    public synchronized String getMessage(String key) throws
InterruptedException {
        while (!m_messages.containsKey(key)) {
            wait();
        }

        return m_messages.remove(key);
    }
}

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

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

    public void run() {
        String key = service.readMessage();

        String msg = null;
        try {
            msg = server.getMessage(key);
            service.sendMessage(msg);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Klasa Service je data na vezbama.