Rešenja prvog kolokvijuma iz Operativnih sistema 2 Januar 2020.

1. (10 poena)

```
class Scheduler {
public:
  Scheduler (): active(0) {}
  PCB* get ();
  void put (PCB*, bool timeExpired=false);

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

  ProcList ready[2][MaxPri+1];
  int active;
};

inline void Scheduler::ProcList::put (PCB* p) {
  if (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 timeExpired=false) {
  int set = (timeExpired?(1-active):active);
  ready[set][p->pri].put(p);
}

PCB* Scheduler::get () {
  int oldActive = active;
  do {
    for (int i=0; i<=MaxPri; i++) {
      PCB* p = ready[active][i].get();
      if (p) return p;
    }
    active = 1-active;
  } while (active!=oldActive);
  return 0;
}
```

2. (10 poena)

```
monitor Agent;
  export takeRedAndGreen,
         takeGreenAndBlue,
         takeRedAndBlue,
  var
    redAvailable, greenAvailable, blueAvailable : boolean;
    waitRG, waitGB, waitRB : condition;

  procedure takeRedAndGreen ();
  begin
    if not (redAvailable and greenAvailable) then
      waitRG.wait();
    redAvailable := false;
    greenAvailable := false;
    putTokens();
  end;

  procedure takeGreenAndBlue ();
  begin
    if not (greenAvailable and blueAvailable) then
      waitGB.wait();
    greenAvailable := false;
    blueAvailable := false;
    putTokens();
  end;

  procedure takeRedAndBlue ();
  begin
    if not (redAvailable and blueAvailable) then
      waitRB.wait();
    redAvailable := false;
    blueAvailable := false;
    putTokens();
  end;

  procedure procedure putTokens ();
  begin
    ...  (* Randomly select two colors and put tokens on the table
            by setting two Boolean variables to True *) ();
    if redAvailable and greenAvailable then
      waitRG.signal();
    if greenAvailable and blueAvailable then
      waitGB.signal();
    if redAvailable and blueAvailable then
      waitRB.signal();
  end;
begin
  putTokens();
end;
```

3. (10 poena)

```
public class Server {
    private final static int N = 10;
    private final ServerSocket socket;
    private int numOfClients = 0;

    public Server() throws IOException {
        socket = new ServerSocket(5555);
    }

    public void work() throws IOException {
        while(true) {
            Socket client = socket.accept();

            new RequestHandler(client, this).start();
        }
    }

    public synchronized int addClient() {
        numOfClients++;
        notifyAll();
        return numOfClients + N;
    }

    public synchronized void waitNewUsers(int count) {
        while(count > numOfClients) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class RequestHandler extends Thread {
    private final Socket client;
    private final Server server;

    public RequestHandler(Socket client, Server server) {
        this.client = client;
        this.server = server;
    }

    public void run() {
        Service service = new Service(client);
        String msg = service.receiveMessage();

            if (msg.equals("Login")) {
                int count = server.addClient();
                server.waitNewUsers(count);
                service.sendMessage("Continue");

            }
    }
}
```

Klasa `Service` (`Usluga`) je data na vežbama.