Rešenja prvog kolokvijuma iz Operativnih sistema 2 novembar 2022.

1. (10 poena)

```
class Scheduler {
public:
  Scheduler () {}
  PCB* get ();
  void put (PCB*, bool wasBlocked, ulong execTime=0);

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

  ProcList ready[MaxPri+1];
};

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, unsigned long execTime) {
  if (pcb==0) return; // Exception!
  if (wasBlocked)
    pcb->execTime = 0;
  else
    pcb->execTime += execTime;
  unsigned long pri = pcb->execTime/TIME_PRI_RESOL;
  if (pri>MaxPri) pri = MaxPri;
  ready[pri].put(p);
}

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

2. (10 poena)

a)(3) Ukoliko izvršavanje operacija `mutex->signal()` i `cond.wait()` ne bi bilo atomično, bilo bi moguće utrkivanje (*race condition*): nit koja je pozvala `Condition::wait()` bi uradila `mutex->signal()` i time otvorila ulaz u operaciju monitora, a onda bi neka druga nit mogla ući u operaciju monitora, izmeniti stanje monitora i uslov čekanja na toj uslovnoj promenljivoj, pa čekanje na uslovu više nije ispravno. Tako bi se prva nit neispravno i nepotrebno blokirala na `cond.wait()`.

b)(7) Razlike i nedostaci u odnosu na semantiku klasičnih uslovnih promenljivih:

- Nije obezbeđeno ograničenje da se uslovna promenljiva (objekat klase `Condition`) može ugraditi samo u objekte monitore.
- Nije obezbeđeno ograničenje da se operacije uslovne promenljive mogu pozivati samo iz konteksta niti koja trenutno izvršava neku operaciju monitora.
- Nije obezbeđeno prioritiranje nastavka izvršavanja niti koja je izvršila operaciju *signal* na uslovnoj promenljivoj u odnosu na niti koje čekaju da tek uđu u operaciju monitora (čekaju na semaforu `mutex`).
- Ako nisu obezbeđena navedena ograničenja, odnosno ukoliko korisnik nije obezbedio međusobno isključenje operacija monitora, i između `if (cond.val()<0)` i `cond.signal()` u `Condition::signal()` može doći do utrkivanja.

3. (10 poena)

```
public class Server {
    private ServerSocket serverSocket = null;
    private final Map<String, List<String>> texts = new HashMap();

    public void doWork() {
        try {
            serverSocket = new ServerSocket(5555);
            while (true) {
                Socket clientSocket = serverSocket.accept();
                Service clientService = new Service(clientSocket);

                RequestHandler  user  =  new  RequestHandler(clientService,
this);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (serverSocket != null) {
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    public synchronized void addText(String title, String text) {
        List<String> versions = texts.get(title);
        if (versions == null) {
            versions = new ArrayList();
            texts.put(title, versions);
        }
        versions.add(text);
    }

    public synchronized String getText(String title, int version) {
        String ret = null;
        List<String> versions = texts.get(title);
        if (versions != null && versions.size() > version) {
            if (version < 0) {
                ret = versions.get(versions.size() - 1);
            } else {
                ret = versions.get(version);
            }
        }
        return ret;
    }

    public static void main(String[] args) {
        Server server = new Server();

        server.doWork();
    }
}
public class RequestHandler extends Thread {
    private final Service service;

    private final Server server;
```

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

    public void run() {
        String msg;
        try {
            while (true) {
                msg = service.receiveMessage();
                if (msg != null) break;
                String[] request = msg.split("#");
                if (request[1].equals("GET")) {
                    getText(request);
                } else {
                    String text = service.receiveMessage();
                    server.addText(request[2], text);
                }
            }
            service.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void getText(String[] request) {
        String title = request[2];
        int version = -1;
        if (request.length > 4) {
            version = Integer.parseInt(request[3]);
        }
        String text = server.getText(title, version);
        if (text == null) {
            service.sendMessage("#NA#");
        } else {
            service.sendMessage("#OK#");
            service.sendMessage(text);
        }
    }
}
```

Klasa `Service` je ista kao na vežbama.