Rešenja prvog kolokvijuma iz Operativnih sistema 2
Januar 2022.

1. (10 poena)

```
class Scheduler {
public:
  Scheduler ();
  void add (PCB*);
  void remove (PCB*);

  PCB* get ();

private:
  PCB* cursor[MaxPri+1];
};

Scheduler::Scheduler () {
  for (unsigned i=0; i<=MaxPri; i++) cursor[i] = 0;
}

inline void Scheduler::add (PCB* p) {
  unsigned pri = p->priority;
  if (cursor[pri]) {
    p->next = cursor[pri]->next;
    cursor[pri]->next = p;
  } else
    cursor[pri] = p->next = p;
}

inline PCB* Scheduler::remove (PCB* p) {
  unsigned pri = p->priority;
  if (p->next!=p) {
    for (PCB* q = p->next; q!=p; q=q->next)
      if (q->next==p) {
        q->next = p->next;
        break;
      }
    if (cursor[pri]==p) cursor[pri] = p->next;
  } else
    cursor[pri] = 0;
  p->next = 0;
}
```

```
PCB* Scheduler::get () {
  for (unsigned i=0; i<=MaxPri; i++) {
    PCB* p = cursor[i];
    if (p==0) continue;
    if (p->next==p)
      if (p->isRunnable) return p;
      else continue;
    for (PCB* q = p; 1; q=q->next)
      if (q->isRunnable) {
        cursor[i] = q->next;
        return q;
      } else
        if (q->next==p) break;
  }
  return 0;
}
```

2. (10 poena)

```
monitor Allocator;
  export acquire, release;
  var
    isFree : boolean,
    numWaiting : integer,
    waitingRoom : condition;

  function acquire : boolean;
  begin
    if isFree then
      begin
        isFree:=false;
        return true;
      end;
    if numWaiting=MaxWaiting then
      return false;
    numWaiting := numWaiting+1;
    waitingRoom.wait();
    return true;
  end;

  procedure release ;
  begin
    if numWaiting=0 then
      isFree:=true;
    else
     begin
       numWaiting := numWaiting - 1;
       waitingRoom.signal();
     end;
  end;

begin
  isFree:=true;
  numWaiting:=0;
end;
```

3. (10 poena)

```
public class Server extends Thread {
    private static final int TIMER_MS = 30000;
    private static final int INIT_PORT = 5555;
    private static final int END_PORT = 6666;

    private int nextPort = INIT_PORT;
    private boolean clientPresent;
    private boolean changePort;
    private boolean samePort;
    private ServerSocket serverSocket = null;

    public void doWork() {
        while (true) {
            try {
                synchronized (this) {
                    serverSocket = new ServerSocket(nextPort);
                    changePort = false;
                }

                nextPort = (nextPort == END_PORT) ? INIT_PORT : nextPort + 1;

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

                    synchronized (this) {
                        clientPresent = true;
                        samePort = !changePort;
                        notify();
                    }

                    Service clientService = new Service(clientSocket);

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

    public void run() {
        while (true) {
            try {
                synchronized (this) {
                    clientPresent = false;
                    wait(TIMER_MS);
                    if (clientPresent) {
```

```
                    continue;
                }
            }
        } catch (InterruptedException e) {
            continue;
        }
        synchronized (this) {
            changePort = true;
            try {
                if (serverSocket != null) {
                    serverSocket.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
            serverSocket = null;
        }
    }

}
}
```