Rešenja prvog kolokvijuma iz Operativnih sistema 2 Septembar 2013.

1. (10 poena)

```
PCB* Scheduler::get () {
  if (maxPri==-1) return idle;
  PCB* ret = head[maxPri];
  head[maxPri] = head[maxPri]->next;
  if (head[maxPri]==0) {
    tail[maxPri]=0;
    while (maxPri>=0 && head[maxPri]==0) maxPri--;
  }
  ret->next = 0;
  return ret;
}
```

2. (10 poena)

```
monitor account;
  export credit, debit;

  var balance : real,
      solvent : condition;

  procedure credit (amount : real);
  begin
    balance := balance + amount;
    solvent.signal();
  end;

  procedure debit (amount : real);
  begin
    while (balance<amount) do solvent.wait();
    balance := balance - amount;
  end;

begin
  balance := 0;
end; (* account *)
```

3. (10 poena)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Server {
 public static final int N = ...;
 public static final int [][] a ={...};
 public static List<Integer> bagOfTasks = Collections.synchronizedList(new
LinkedList<Integer>());
 public static int result;
 public static Worker[] workers;
 public static int workerNum;
 public static void main(String[] args) {
   ServerSocket sock = new ServerSocket(1033);
   while (true) {
    Socket clientSocket = sock.accept();
    BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
    PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),true);
```

```
    while (true) {
     String request = in.readLine();
     StringTokenizer st = new StringTokenizer(request, "#");
     String functionName = st.nextToken();
     if (functionName.equals("start")) {
      workerNum = Integer.parseInt(st.nextToken());
      result=0;
      out.println(""+N);
     } else if (functionName.equals("calculate")){
      for (int i = 0; i <= st.countTokens(); i++)
       bagOfTasks.add(Integer.parseInt(st.nextToken()));

      workers = new Worker[workerNum];
      for (int i = 0; i < workers.length; i++) {
       workers[i]= new Worker();
       workers[i].start();
      }
      synchronized (Server.class) {
       while (workerNum>0){ ; //wait to complete all
        Server.class.wait();
       }
      }
      out.println(""+result);
      break;
     }
    }
   }
 }
}

class Worker extends Thread{
 @Override
 public void run() {
  while (!Server.bagOfTasks.isEmpty()) {
   int i = Server.bagOfTasks.remove(0), res=0;
   for (int j = 0; j < Server.N; j++) {
    res+=Server.a[i][j];
   }
   synchronized (Server.class) {
    Server.result+=res;
   }
  }
  synchronized (Server.class) {
   Server.workerNum--;
   Server.class.notifyAll();
  }
 }
}
```