Rešenja prvog kolokvijuma iz Operativnih sistema 2 decembar 2025.

1. (10 poena) 5, 6, 7, 6, 7, 6, 5, 6, 5, 4

2. (10 poena)

```
class ReaderWriters {
public:
  ReaderWriters () {}
  const T* read () const;
  void write (const T&);

private:
  T data[2];
  int toRead = -1, toWrite = -1;
  Semaphore mutex = 1;
};

void ReaderWriters::write (const T& t) {
  mutex.wait();
  if (toWrite<0) toWrite = 0;
  data[toWrite] = t;
  mutex.signal();
}

const T* ReaderWriters::read (){
  mutex.wait();
  if (toWrite<0) { mutex.signal(); return 0; }
  toRead = toWrite;
  toWrite = 1 - toWrite;
  T* ret = &data[toRead];
  mutex.signal();
  return ret;
}
```

3. (10 poena)

```
public class ClientCalculateRPC extends Thread {
    private Service service;
    private boolean work = true;
    private boolean successful;

    public ClientCalculateRPC(Service service) {
        this.service = service;
        start();
    }

    public void setData(int[] data) throws Exception {
        service.sendMsg("SET_DATA#" + data.length);
        for (int value : data) {
            service.sendMsg(String.valueOf(value));
        }
        startMeasurement();
        service.receiveMsg();
        finishMeasurement();

    }

    private synchronized void startMeasurement() {
        successful = false;
        notify();
    }

    private synchronized void finishMeasurement() {
        successful = true;
        notify();
    }

    public int[] calculate() throws Exception {
        service.sendMsg("CALCULATE");
        startMeasurement();
        int len = Integer.parseInt(service.receiveMsg());
        int[] result = new int[len];
        for (int i = 0; i < len; i++) {
            result[i] = Integer.parseInt(service.receiveMsg());
        }
        finishMeasurement();
        return result;
    }

    public void stopWork() {
        work = false;
        service.stop();
    }

    public void run() {
```

```
        while (work) {
            synchronized (this) {
                try {
                    wait();
                    wait(10000);
                    if (!successful) {
                        System.out.println("Operation timed out");
                        work = false;
                        service.close();
                    }

                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }

        }

    }
}
```