Rešenja zadataka za drugi kolokvijum iz Operativnih sistema 1 Jun 2018.

1. (10 poena)

```
const int N = ...;  // Capacity of the buffer

class BoundedBuffer {
public:

  BoundedBuffer ();

  void  put1 (int);
  void  put2 (double);
  int   get  (int*, double*);

private:
  Semaphore mutex;
  Semaphore spaceAvailable1, itemAvailable1;
  Semaphore spaceAvailable2, itemAvailable2;

  int buffer1[N];
  int head1, tail1;

  double buffer2[N];
  int head2, tail2;
};

BoundedBuffer::BoundedBuffer () :
  mutex(1),
  spaceAvailable1(N), itemAvailable1(0),
  head1(0), tail1(0),
  spaceAvailable2(N), itemAvailable2(0),
  head2(0), tail2(0)  {}

void BoundedBuffer::put1 (int d) {
  spaceAvailable1.wait();
  mutex.wait();
  buffer1[tail1] = d;
  tail1 = (tail1+1)%N;
  mutex.signal();
  itemAvailable1.signal();
}

// Similar for put2

int BoundedBuffer::get (int* pi, double* pd) {
  int s = Semaphore::waitOr(&itemAvailable1,&itemAvailable2);
  mutex.wait();
  if (s==1) {
    *pi = buffer1[head1];
    head1 = (head1+1)%N;
    mutex.signal();
    spaceAvailable1.signal();
  } else {
    *pd = buffer2[head2];
    head2 = (head2+1)%N;
    mutex.signal();
    spaceAvailable2.signal();
  }
  return s;
}
```

2. (10 poena)

a)(4)

Zapis broj	Adresa početka	Veličina
1	34B0	10
2	3510	20
3	3680	30
4	35A0	90

b)(3)

Zapis broj	Adresa početka	Veličina
1	34B0	10
2	3680	30
3	3510	120

c)(3)

Zapis broj	Adresa početka	Veličina
1	3550	160

3. (10 poena) a)(3) VA(32): Page1(9):Page2(9):Offset(14).
PA(24): Frame(10):Offset(14).

b)(7)

```
const unsigned offsw = 14;
enum AddrExcKind {pageFault, opDenied};

int resolveVAddrExc (PCB* pcb, unsigned long vaddr, AddrExcKind kind,
                     short rwx, SegDesc** ret) {
  if (pcb==0) return ERR_FATAL; // Fatal exception!
  unsigned long page = addr>>offsw;
  for (SegDesc* sd = pcb->segDesc; sd!=0; sd = sd->next) {
    if (page>=sd->startingPage && page<sd->startingPage+sd->size) {
      *ret = sd;
      switch (kind) {
        case pageFault: return LOAD_PAGE;
        case opDenied:
          if ((rwx&2) && (sd->rwx&2))
            return COPY_ON_WRITE;
          else
            return MEM_ACCESS_FAULT;
      }
    }
  }
  *ret = 0;
  return MEM_ACCESS_FAULT;
}
```