Kolokvijum iz Operativnih sistema 1
jul 2021.

1. (10 poena)

```
const size_t N = ...;

int ai[N], bj[N], ci[N];

void mat_mul () {
  for (size_t i=0; i<N; i++) {
    load_arow(&ai,i);
    for (size_t j=0; j<N; j++) {
      load_bcol(&bj,j);
      ci[j] = 0;
      for (size_t k=0; k<N; k++)
        ci[j] += ai[k]*bj[k];
    }
    store_crow(&ci,i);
  }
}
```

2. (10 poena)

```
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <unistd.h>  // for fork()
#include <stdio.h> // for printf

sem_t *sem1, *sem2;

int main () {
  sem1 = sem_open("/sem1", O_CREAT|O_EXCL, O_RDWR, 1);
  if (sem1 == NULL) return -1;
  sem2 = sem_open("/sem2", O_CREAT|O_EXCL, O_RDWR, 0);
  if (sem2 == NULL) { sem_close(sem1); return -1; }

  pid_t pid = fork();
  if (pid<0) return -1;

  for (int i=0; i<10; i++)
    if (pid>0) {
      if (sem_wait(sem1)<0) break;
      printf("A"); fflush(stdout);
      if (sem_post(sem2)<0) break;
    } else {
      if (sem_wait(sem2)<0) break;
      printf("B"); fflush(stdout);
      if (sem_post(sem1)<0) break;
    }

  sem_close(sem1); sem_unlink("/sem1");
  sem_close(sem2); sem_unlink("/sem2");

  return 0;
}
```

3. (10 poena)

```
interrupt void packetArrived () {
  if (pbTail == pbHead) {  // Buffer full, reject packet
    *ioCtrl = IO_REJECT;
  } else {
    for (int i=0; i<PACKET_SIZE; i++) pbTail[i] = ioData[i];
    *ioControl = IO_COMPLETE;
    if ((pbTail-packetBuffer)/PACKET_SIZE == BUFFER_SIZE - 1)
      pbTail = packetBuffer;
    else
      pbTail = pbTail+PACKET_SIZE;
  }
}
```

4. (10 poena)

```
int extendFile (FCB* fcb) {
  size_t entry = (fcb->size+BLOCK_SIZE-1)/BLOCK_SIZE;
  if (entry<SingleIndexSize) {
    PBlock newBlock = allocateBlock();
    if (!newBlock) return -1;  // Error: no free space for the new block
    fcb->singleIndex[entry] = newBlock;
    return 0;
  }

  entry -= SingleIndexSize;
  if (entry>=DblIndexSize) return -2;  // Error: file size limit exceeded

  if (!fcb->dblIndex) {
    fcb->dblIndex = allocateBlock();
    if (!fcb->dblIndex) return -3; // Error: no space for index block
  }

  PBlock newBlock = allocateBlock();
  if (!newBlock) return -1;  // Error: no free space for the new block
  PBlock* dblIx = (PBlock*)getBlock(fcb->dblIndex);
  dblIx[entry] = newBlock;
  return 0;
}
```