Rešenja drugog kolokvijuma iz Operativnih sistema 2, januar 2019.

1. (10 poena) a)(7) Da bi se proverilo da li mrtva blokada sigurno postoji, treba eliminisati sve procese koji mogu dobiti tražene resurse i potom možda osloboditi one koje već drže:

Allocation

	A	*B*	*C*
*P*1	1	0	1
*P*2	2	1	1
*P*3	0	1	1
*P*4	0	1	1

Request

	A	*B*	*C*
*P*1	0	1	3
*P*2	0	0	0
*P*3	2	0	1
*P*4	3	1	0

Available

A	*B*	*C*
0	0	0

Proces *P2* ne traži resurse, pa se može dogoditi da on oslobodi svoje zauzete resurse:

Allocation

	A	*B*	*C*
*P*1	1	0	1
~~*P*2~~	~~2~~	~~1~~	~~1~~
*P*3	0	1	1
*P*4	0	1	1

Request

	A	*B*	*C*
*P*1	0	1	3
~~*P*2~~	~~0~~	~~0~~	~~0~~
*P*3	2	0	1
*P*4	3	1	0

Available

A	*B*	*C*
2	1	1

Sada proces *P3* može da dobije tražene resurse, jer je za njega $Request_3 \leq Available$, pa se može dogoditi da on oslobodi svoje resurse:

Allocation

	A	*B*	*C*
*P*1	1	0	1
~~*P*2~~	~~2~~	~~1~~	~~1~~
~~*P*3~~	~~0~~	~~1~~	~~1~~
*P*4	0	1	1

Request

	A	*B*	*C*
*P*1	0	1	3
~~*P*2~~	~~0~~	~~0~~	~~0~~
~~*P*3~~	~~2~~	~~0~~	~~1~~
*P*4	3	1	0

Available

A	*B*	*C*
2	2	2

Međutim, sada ni proces *P1* ni proces *P4* ne može da dobije tražene resurse jer ni za jedan od njih ne važi $Request_i \leq Available$, pa su ova dva procesa sigurno u mrtvoj blokadi.

b)(3) Može se ugasiti bilo koji od ova dva procesa i mrtva blokada će biti rešena (lako se proverava na opisani način). Da to nije tako, tj. ako bi ostao jedan proces koji ne može zadovoljiti svoje zahteve, to bi značilo da su njegovi zahtevi za resursima veći od ukupno raspoložive količine resursa u sistemu, pa on svakako ne može da se izvršava čak ni sam.

2. (10 poena)

```
const uint16 PMTSize = ((uint16)1)<<10;
typedef uint32 PgDesc[2];

PgDesc* _getVictimPageDesc (uint32* pmt, uint8 flags) {
  for (uint16 i=0; i<PMTSize; i++) {
    PgDesc* pmt1 = (PgDesc*)(pmt[i]);
    if (pmt1==0) continue;
    for (uint16 j=0; j<PMTSize; j++) {
      uint32 pgDesc = pmt1[j][0];
      if ((pgDesc>>30)==flags) return &pmt1[j];
    }
  }
  return 0;
}

PgDesc* getVictimPageDesc (uint32* pmt) {
  if (pmt==0) return 0; // Exception
  PgDesc* victim = 0;

  victim = _getVictimPageDesc(pmt,0);
  if (victim) return victim;

  victim = _getVictimPageDesc(pmt,2);
  if (victim) return victim;

  victim = _getVictimPageDesc(pmt,1);
  if (victim) return victim;

  victim = _getVictimPageDesc(pmt,3);
  if (victim) return victim;

  return 0; // Exception: there are no pages of this process
}
```

3. (10 poena)

Dati program ispisuje deo sadržaja fajla zadatog svojim prvim argumentom iz komandne linije na standardni izlaz. Opseg bajtova sadržaja fajla koji se ispisuju zadaje se pomerajem (*offset*) i dužinom (*length*) u drugom i trećem argumentu komandne linije (ako treći argument nije zadat ili prebacuje veličinu fajla, ispisuje se sadržaj do kraja fajla). Program pravi memorijsko preslikavanje sadržaja fajla potrebne veličine u stranicama (i poravnat na stranicu) i potom ispisuje tražene bajtove na standardni izlaz sistemskim pozivom *write.*