Rešenja drugog kolokvijuma iz Operativnih sistema 2, februar 2025.

1. (10 poena)

a)(5) Pošto nema ugnežđenih zaključavanja kritičnih sekcija osim ako se pozivaju potprogrami iz drugog modula, krug držanja i čekanja kritičnih sekcija, kao neophodan uslov za nastanak mrtve blokade, mogao bi se zatvoriti samo ako procedura iz jednog modula indirektno poziva proceduru iz istog modula koja zahteva isti ključ (semafor) koji ova prva već drži. To bi značilo da graf zavisnosti modula mora da sadrži petlju, pa je postojanje petlje neophodan uslov da bi mrtva blokada bila moguća, odnosno nepostojanje petlje je dovoljan uslov da ona nije moguća. Ovaj uslov nije potreban, jer čak i ako postoji petlja zavisnosti između modula, jedna procedura ne mora indirektno pozivati proceduru iz istog modula koja zahteva isti ključ ili uopšte zaključava nešto.

b)(5) Procedure istog modula koje koriste isti ključ (semafor) čine istu klasu potprograma i tu klasu treba predstaviti istim čvorom, a različite klase različitim čvorovima grafa. Usmerene grane grafa treba da označavaju poziv nekog od potprograma iz jedne klase od strane nekog potprograma iz druge klase. Tada je uslov da je ovaj graf DAG potreban i dovoljan za nemogućnost nastanka mrtve blokade.

2. (10 poena)

```
size_t getVictimPage (PMT0_t* pmt0) {
  unsigned int minb = (unsigned int)-1;
  size_t minp = (size_t)-1;
  for (size_t i=0; i<PMT0_size; i++) {
    PMT1_t* p = (*pmt0)[i];
    if (p==0) continue;
    size_t ix = p-pmt1;
    for (size_t j=0; j<PMT1_size; j++)
       if (!p[j] && refBits[ix][j]<minb) {
         minb = refBits[ix][j];
         minp = i*PMT1_size+j;
       }
  }
  return minp;
}
```

3. (10 poena)

```
int calculateSlabSize(int pageSize, int objectSize, int metaDataSize, int
maxSlabSize)
{
    int bestN = 1;
    int bestFrag = objectSize;

    for (int n = 1; n <= maxSlabSize; n++) {
        int usable = n * pageSize - metaDataSize;
        if (usable <= 0)
            continue;
        int frag = usable % objectSize;
        if (frag < bestFrag) {
            bestFrag = frag;
            bestN = n;
        }
        if (bestFrag == 0)
            break;
    }
```

```
    return bestN;
}
```