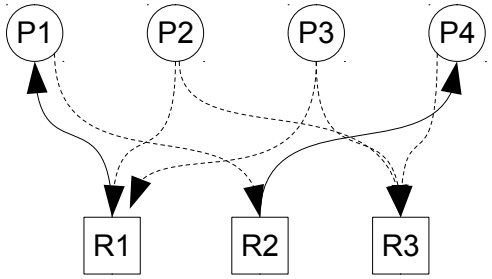
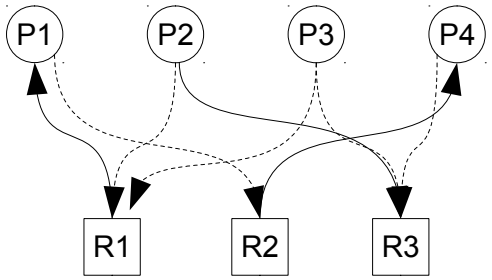
Rešenja drugog kolokvijuma iz Operativnih sistema 2, decembar 2019.

1. (10 poena)

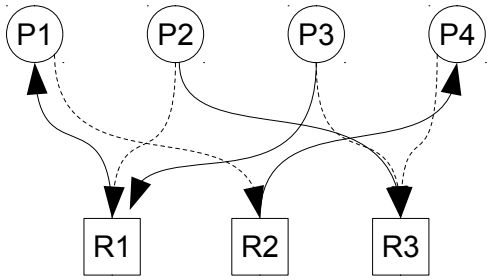
a)(2)



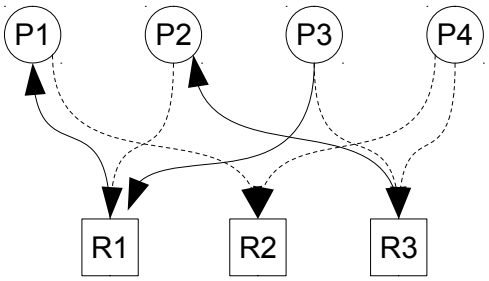
b)(3)



c)(2)



d)(3)



2. (10 poena)

```
void scanPages (PMT0 pmt0, uint32* victim) {
  uint8 minref = 0xff;
  for (uint32 p0=0; p0<PMT0size; p0++) {
    if (pmt0[p0]==0) continue;
    PMT1* pmt1 = (PMT1*)pmt0[p0];
    for (uint32 p1=0; p1<PMT1size; p1++) {
      uint32 val = (*pmt1)[p1][1];
      uint8 ref = val & 0xff;
      if (victim && ((*pmt1)[p1][0]!=0) && (ref<minref)) {
        minref = ref;
        *victim = (p0<<10) | p1;
      }
      ref >>= 1;
      ref |= ((uint8)(val>>24)) & 0x80);
      val = (val & ref) | ref;
      (*pmt1)[p1][1] = val;
    }
  }
}
```

3. (10 poena)

a)(5) Garantovano najmanji broj straničnih grešaka pod navedenim uslovima pri navedenom obilasku stabla, za bilo koji broj raspoloživ okvira za smeštanje tog stabla (pa i za jedan jedini okvir), biće u slučaju da obilazak stabla pristupa čvorovima tako da taj redosled odgovara sekvencijalnom prolasku kroz elemente niza u kom su čvorovi, tj. onda kada redosled pristupa čvorovima odgovara sekvencijalnom pristupu svim elementima niza, redom. Zato taj niz treba prepakovati upravo tako, da istim navedenim redosledom obilaska stabla u niz budu poređani i čvorovi tog stabla (po rastućem redosledu indeksa). Tada će obilazak stabla pristupati jednoj po jednoj stranici na koje se prostire niz, i to sekvencijalno, pri čemu će jednoj stranici pristupati samo jednom i nikada ponovo (upravo zbog toga je urađeno fino doterivanje koda petlje opisano u tekstu zadatka), pa će broj straničnih grešaka biti jednak broju stranica koje zauzima dati niz (to je i teorijski minimum).

b)(5)

```
void repack (Node oldTree[], Node newTree[], unsigned oldNode) {
  newTree[size] = oldTree[oldNode];
  newTree[size].child = (oldTree[node].child?(size+1):0);
  size++;
  for (unsigned i=oldTree[oldNode].child; i!=0; i=oldTree[i].sibling) {
    unsigned newNode = size;
    repack(oldTree,newTree,i);
    newTree[newNode].sibling = (oldTree[i].sibling?size:0);
  }
}
```