Rešenja zadataka za kolokvijum iz Operativnih sistema 1 – Oktobar 1 2025.

1. (10 poena)

```
void* createRegion (RegionDesc* phead, size_t sz) {
  if (!phead || !sz) return nullptr;

  RegionDesc* bestPrev = nullptr;
  size_t bestSize = 0;
  for (RegionDesc* prev = phead; prev; prev = prev->next) {
    size_t freeSz;
    if (prev->next) freeSz = prev->next->addr – (prev->addr+prev->size);
    else freeSz = MAX_VADDR+1 – (prev->addr+prev->size);
    if ((freeSz>=sz) && (!bestPrev || freeSz<bestSize)) {
      bestPrev = prev;
      bestSize = freeSz;
    }
  }

  if (!bestPrev) return nullptr;

  RegionDesc* dsc = (RegionDesc*)kmalloc(sizeof(RegionDesc));
  if (!dsc) return nullptr;
  size_t addr = bestPrev->addr+bestPrev->size;
  dsc->addr = addr; dsc->size=sz; dsc->next = bestPrev->next;
  bestPrev->next = dsc;
  return addr;
}
```

2. (10 poena)

```
template<typename T, int PackageSize, int NumOfPkgs>
class BoundedBuffer {
public:
  BoundedBuffer () {}
  void append (T pkg[]);
  T take ();

private:
  static const int size = PackageSize*NumOfPkgs;
  T buffer[size];
  int head = 0, tail = 0;
  Semaphore mxProd=1, mxCons=1, spaceAvailable=size, itemAvailabe=0;
};
```

```
template<typename T, int P, int N>
void BoundedBuffer<T,P,N>::append (T p[]) {
  for (int i=0; i<P; i++) spaceAvailable.wait();
  mxProd.wait();
    for (int i=0; i<P; i++) {
      buffer[tail] = p[i];
      tail = (tail+1) % size;
    }
  mxProd.signal();
  for (int i=0; i<P; i++) itemAvailable.signal();
}
```

```
template<typename T, int P, int N>
T BoundedBuffer<T,P,N>::take () {
  itemAvailable.wait();
  mxCons.wait();
    T t = buffer[head];
    head = (head+1) % size;
  mxCons.signal();
  spaceAvailable.signal();
  return t;
}
```

3. (10 poena)

a)(5) Čvorovi koji se smeštaju u isti blok su oni sa indeksima $2k$ i $2k+1$, gde je k neki prirodan broj ili 0. Postoje dve mogućnosti:

i) Ova dva čvora pripadaju istom nivou stabla (susedni *siblings*). Tada ova dva čvora imaju različite roditelje: k-1, odnosno k. Roditelju prvog čvora taj čvor je desno dete, a roditelju drugog čvora taj čvor je levo dete. Kako se deca čvorova uvek obilaze nakon roditelja, posmatrana dva čvora nikada neće biti posećena kao susedna, pa će sa obilazak svakoga morati da se učita blok u kome se taj čvor nalazi. Dakle, za svaki ovakav čvor mora se učitati blok iznova.

ii) Prvi čvor je krajnje desni u nivou i, a drugi čvor je prvi levi u nivou $i+1$. Ni ova dva čvora se nikada ne obilaze jedan za drugim, osim u početnom slučaju kad je $k=0$, odnosno za koren i njegovo levo dete.

Prema tome, u ovom slučaju samo za čvor sa indeksom 1 neće biti učitan novi blok, za svaki drugi čvor mora se učitati blok. Zato je ukupno potrebno učitati 2^n-2 blokova, za $n > 1$, odnosno $\max(2^n-2, 1)$ blokova.

b)(5) Čvorovi se sada obilaze tačno po redosledu indeksa u nizu, pa će za svaka dva čvora biti potrebno učitati nov blok (plus jedan zbog neparnog broja čvorova). Zato je ukupno potrebno učitati $(2^n-1+1)/2 = 2^{n-1}$ blokova.

4. (10 poena)

```
void setAccessBits (const char str[4], short oldBits, short& newBits) {
  const char mod = str[0];

  newBits = oldBits;
  if (mod=='=') newBits &= ~(short)0b111;

  for (int i=1; i<=3 && str[i]; i++)
    switch (str[i]) {
      case 'r', 'R':
        if (mod=='=' || mod=='+') newBits |= 0b100;
        if (mod=='-') newBits &= 0b011;
        break;

      case 'w', 'W':
        if (mod=='=' || mod=='+') newBits |= 0b010;
        if (mod=='-') newBits &= 0b101;
        break;

      case 'x', 'X':
        if (mod=='=' || mod=='+') newBits |= 0b001;
        if (mod=='-') newBits &= 0b110;
        break;
    }
}
```