JMBAG **IME I PREZIME**

Algoritmi i strukture podataka –1. Ispitni rok

1.srpnja 2014.

*Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe **goto**. Neefikasna rješenja mogu donijeti manje bodova.Nerekurzivne funkcije se ne priznaju kao rješenja u zadacima u kojima se traži rekurzivna funkcija i obratno.*
*Ispit nosi maksimalno 70 bodova, a prag za prolaz pismenog ispita je **35** bodova. 3. zadatak rješavate na ovom obrascu dok ostale zadatke rješavate na svojim listovima papira. Ovaj obrazac morate predati.*

Zadatak 1. (15 bodova)

Neka su zadani tipovi podataka `cvor` koji predstavlja čvor binarnog stabla i `atom` koji predstavlja element reda.

```
typedef struct cv {
        int element;
        struct cv *lijevo, *desno;
} cvor;
```

```
typedef struct at {
        cvor element;
        struct at *sljed;
} atom;
```

Za rad s redom na raspolaganju su funkcije za inicijalizaciju reda, dodavanje i skidanje elementa iz reda. Funkcije za dodavanje i skidanje elementa iz reda vraćaju 1 ako su se izvršile uspješno.

```
void init_red(Red *red);
int dodajURed(cvor element, Red *red);
int skiniIzReda(cvor *element, Red *red);
```

Napišite **nerekurzivnu** funkciju koja će ispisati postojeće **puno** binarno stablo tako da sve čvorove na istoj razini ispisuje u jednom retku, a svaku razinu u novom retku. Prototip tražene funkcije je:

```
void ispisBezRekurzije(cvor* korijen);
```

Npr. za stablo prikazano na lijevoj strani funkcija mora ispisati čvorove stabla na način prikazan na desnoj strani (prvi redak u primjeru ispisa predstavlja redne brojeve stupaca na zaslonu):

```
      10
    /    \
   8      20
  / \    / \
 7   9  15  22
```

```
01234567890123456789
  10
   8  20
   7   9  15  22
```

Napomene:
U implementaciji tražene funkcije može se koristiti red (ili više njih). Za rad s redom dozvoljeno je koristiti isključivo navedene funkcije. Rekurzivna rješenja neće se priznati.

Zadatak 2. (15 bodova)

U jednostruko povezanu nesortiranu listu spremljeni su podaci o studentima. Lista je zadana sljedećim strukturama:

```
struct zapis {
    char imeprezime[80+1];
    int visina;
};
```

```
struct at {
    struct zapis element;
    struct at *sljed;
};
```

Napišite funkciju čiji je prototip:

```
int rasporediElemente(atom** glava);
```

koja će elemente liste (studente) rasporediti ovisno o odnosu njihove visine i visine studenta koji je u početnoj listi na početku (početni student). U novoj listi, oni studenti koji su viši od početnog studenta trebaju se naći s njegove „lijeve“ strane (bliže početku) u obrnutom redosljedu od početnog, dok se ostali studenti trebaju naći s njegove „desne“ strane u originalnom redosljedu. Funkcija vraća novu poziciju elementa koji je u početnoj listi bio na početku (na poziciji 1).

Npr. ako su visine studenata u početnoj listi **169**, 196, 160, 161, 171, 197, 158, nakon raspoređivanja trebaju biti 197, 171, 196, **169**, 160, 161, 158 (početak liste je na studentu čija je visina 197). Funkcija treba vratiti 4.

Zadatak 3. (15 bodova)

Zadana je dvostruko povezana lista atoma definiranih odsječkom:

```
typedef struct at {
       int vrijednost;
       struct at *slijed;
       struct at *preth;
} atom;
```

Atomi su u listi sortirani uzlazno prema vrijednosti: prvi atom (čiji je pokazivač preth jednak NULL) ima najmanju vrijednost. Dopunite definiciju funkcije brisi koja iterira po listi od prvog elementa prema zadnjem tražeći atom zadane vrijednosti. Ako nađe takav atom, vadi ga iz liste i vraća njegovu adresu, a ako ne nađe takav atom vraća NULL.

```
atom * brisi(atom **glava, int vrijednost) {
  int prvaiteracija = 1;
  atom *pom;
  while(*glava) {
    if(_____________________________> vrijednost) {
      return NULL;
    }
    if(_____________________________ == vrijednost ) {
      if(_____________________________ != NULL) {
        _____________________________ = _________________;
      }
      if_____________________________ != NULL) {
        _____________________________ = _________________;
      }
      pom = *glava;
      if(prvaiteracija) {
        _____________ = _____________________________;
      }
      return pom;
    }
    prvaiteracija = 0;
    _________________ = _____________________________;
  }
  return NULL;
}
```

Zadatak 4. (10 bodova)

U polje cjelobrojnih brojeva pohranjen je sljedeći niz brojeva:

4, 6, 7, 2, 9, 8, 1, 5, 3

a) **(5 bodova)** Ilustrirajte uzlazno sortiranje algoritmom mergesort. Potrebno je prikazati sadržaj polja nakon svake promjene.Pri particioniranju neparnog broja elemenata neka bude desna particija veća. Redosljed izmjena je bitan: nije dopušteno paralelno izvršavanje operacija, a lijeve particije se obrađuju prije desnih (mergesort je implementiran rekurzivnom funkcijom koja radi u jednoj dretvi odnosno *thread-u*).

b) **(5 bodova)** Ilustrirajte uzlazno sortiranje algoritmom quicksort. Stožer za quicksort birajte metodom aproksimacije medijana temeljem prvog, srednjeg i zadnjeg člana, a vrijedi da je `cutoff = 3`, nakon čega `se sortira bez navođenja koraka`.

Zadatak 5. (15 bodova)

Jedan zapis datoteke organizirane po načelu raspršenog adresiranja sadrži podatke o tjednim promjenama cijena proizvoda: šifru (int), naziv (50+1 znak) i promjenu cijene (float) proizvoda u odnosu na protekli tjedan, te godinu (int) i tjedan u godini (int) na koji se promjena odnosi. Prazni se zapis prepoznaje po šifri jednakoj nula (0). Veličina bloka na disku je 2048 B. Očekuje se najviše 50000 zapisa, a tablica je dimenzionirana za 35% veći kapacitet. Prilikom upisa primjenjuje se metoda cikličkog preljeva. Poznato je da nakon što se zapis jednom upiše, nema više brisanja zapisa. Ključ zapisa je kombinacija šifre, godine i tjedna, a pretvorba ključa u adresu se obavlja već pripremljenom funkcijom

```
int adresa(int sifra, int godina, int tjedan);
```

Napišite sve potrebne deklaracije (strukturu i parametre za hash).

Napisati funkciju koja će za svaki pretinac odrediti najveću promjenu cijena proizvoda u tom pretincu, ali samo među onim proizvodima koji su u taj pretinac izvorno upućeni (nisu u preljevu) i potom tu ocjenu ispisati u formatu

```
Pretinac broj_pretinca: najveća promjena cijena
```

a ako takva cijena ne postoji, nije potrebno ispisati ništa. Funkcija treba imati prototip:

```
void fun(FILE *f);
```

Rješenja:

Zadatak 1. (15 bodova)

```
void ispisBezRekurzije(cvor* korijen) {
Red red;
cvor cv;
int razina = 1, brojCvorova = 0;

if(korijen == NULL) {
  return;
}

init_red(&red);
dodajURed(*korijen, &red);

while(skiniIzReda(&cv, &red)) {
    printf("%4d ", cv.element);
    brojCvorova++;
    if(cv.lijevo) {
      dodajURed(*(cv.lijevo), &red);
    }
    if(cv.desno) {
      dodajURed(*(cv.desno), &red);
    }
    if(brojCvorova >= pow(2, razina - 1)) {
      razina++;
      brojCvorova = 0;
      printf("\n");
    }
  }
}
```

Zadatak 2. (15 bodova)

```
int rasporedi(atom **glava){
  atom *prviVeci,*zadnjiManji, *trenutni, *slijedeci;
  int novaPozicija;
  double vrijednost;

  if(!(*glava)) return 0;

  vrijednost=(*glava)->element.visina; // visina početnog studenta
  novaPozicija=1;
  prviVeci=zadnjiManji=*glava; // pamtim prvog većeg i zadnjeg manjeg
  trenutni=(*glava)->sljed;
  zadnjiManji->sljed=NULL;  // zadnji manji pokazuje na kraj
  while(trenutni){ // dok ima elemenata u listi
    slijedeci=trenutni->sljed; // zapamtim slijedećeg
    if(trenutni->element.visina > vrijednost){ // ako je viši od početnog
      trenutni->sljed=prviVeci; // stavim ga na početak liste
      prviVeci=trenutni;
      novaPozicija++; // i zapamtim novu poziciju
     }
    else{
      zadnjiManji->sljed=trenutni;  // inače ga stavim na kraj
      zadnjiManji= zadnjiManji->sljed;
      zadnjiManji->sljed=NULL;
    }
    trenutni=slijedeci; // prijeđem na novi element liste
  }
  *glava=prviVeci; // vratim novi početak
  return novaPozicija; // i novu poziciju starog početnog…
}
```

Zadatak 3. (15 bodova)

```
atom * brisi(atom **glava, int vrijednost) {
  int prvaiteracija = 1;
  atom *pom;
  while(*glava) {
    if((*glava)->vrijednost > vrijednost) {
      returnNULL;
    }
    if((*glava)->vrijednost == vrijednost ) {
      if((*glava)->preth != NULL) {
        (* glava)->preth->sljed =  (*glava)->sljed;
      }
      if((*glava)->sljed != NULL) {
        (*glava)->sljed->preth = (*glava)->preth;
      }
      pom = *glava;
      if(prvaiteracija) {
        *glava = (*glava)->sljed;
      }
      return pom;
    }
    prvaiteracija = 0;
    glava = &(*glava)->sljed;
  }
  return NULL;
}
```

Zadatak 4. (10 bodova)

a) (5 bodova)

4	6	**7**	**2**	9	8	1	5	3	
4	**6**	**2**	**7**	9	8	1	5	3	prva izmjena: desna potparticija lijeve particije
2	4	6	7	**9**	**8**	1	5	3	nakon merge-a lijevih particija
2	4	6	7	8	9	1	**5**	**3**	kod desne particije prvo se složi lijeva (manja) potparticija
2	4	6	7	**8**	**9**	**1**	**3**	**5**	zatim desna potparticija (od desne particije)
2	**4**	**6**	**7**	**1**	**3**	**5**	**8**	**9**	pa i pri merge-anju desne ima izmjena
1	2	3	4	5	6	7	8	9	i konačni merge

b)(5 bodova)

4	6	7	2	**9**	8	1	5	**3**	početno
3	6	7	2	**4**	8	1	**5**	9	stožer odabran
3	**6**	7	2	5	8	**1**	4	9	stožer sklonjen
3	1	**7**	**2**	5	8	6	4	9	i=1, j=6 zamijenjeni
3	1	2	**7**	5	8	6	**4**	9	i=2, j=3 zamijenjeni
3	**1**	**2**	4	5	8	6	7	9	i,j se mimoišli, stožer vraćen
1	2	3	4	5	8	**6**	**7**	9	lijevo riješeno insertionom
1	2	3	4	5	**8**	7	**6**	9	stožer desne particije sklonjen
1	2	3	4	5	**6**	7	**8**	9	zamjena u desnom dijelu, zadnja

Zadatak 5. (15 bodova)

```
typedef struct{
       int sifra;
       char naziv[50+1];
       int godina;
       int tjedan;
       float promjena;
} zapis;


void najveca(FILE *f){
  zapis pretinac[C];
  int i, j, adr;
  double najPretinac;
  int imaNesto;

  if (f==NULL) return;

  for (i = 0; i < M; i++) {
    fseek (f, i * BLOK, SEEK_SET);
    fread (pretinac, sizeof (pretinac), 1, f);
    imaNesto=0;
    for (j = 0; j < C; j++) {
      if (pretinac[j].sifra != 0) {
        /* Ako zapis nije prazan */
        if (adresa(pretinac[j].sifra
            , pretinac[j].godina, pretinac[j].tjedan) == i) { // ako je zapis u pravom pretincu ..
          if (imaNesto!=0) { // postoji kandidat za najveću vrijednost
            if(pretinac[j].promjena >najPretinac)
              najPretinac=pretinac[j].promjena;
          }
          else{
            najPretinac=pretinac[j].promjena;
            imaNesto=1;
          }
        }
      }
      else break; // ostatak pretinca je prazan pa ga nije potrebno provjeravati
    }
    if (imaNesto){
      printf("(Pretinac %d) Najveca promjena: %7.2f\n",i,najPretinac);
    }
  }
}
```