Završni ispit iz predmeta Algoritmi i strukture podataka

23. lipnja 2008.

Napomena za sve zadatke:
- Nije dopušteno korištenje naredbe `goto`, te statičkih i globalnih varijabli.

Zadatak 1. *(8 bodova)*

a) Napišite funkciju čiji je prototip

```
int jeligomila(int stablo[], int n);
```

koja će vratiti 1 ako je potpuno binarno stablo s **n** elemenata (implementirano poljem) gomila, a 0 inače.

b) Napišite funkciju čiji je prototip

```
int jeligomila(cvor *korijen);
```

koja će vratiti 1 ako je potpuno binarno stablo (*čiji je korijen zadan*) gomila, a 0 ako nije.

Zadatak 2. *(8 bodova)*

U jednostruko povezanu listu spremaju se podaci o radnicima nekog poduzeća. Lista je zadana sljedećim strukturama.

```
struct zapis {
  char imeprezime[30+1];
  int matbr;
  int godisnji;
};

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

typedef struct at atom;
```

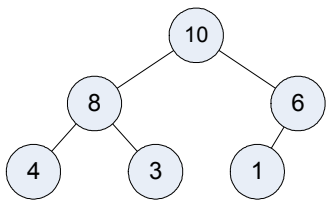
Jedan element liste sadrži sljedeće podatke: ime i prezime radnika (30+1 znakova), matični broj radnika (int) i broj dana godišnjeg odmora (int). Lista nije sortirana. Napisati funkciju koja će iz liste izbaciti one radnike koji imaju više od 30 dana godišnjeg odmora. Funkcija mora imati prototip:

```
void izbaci(atom **glava);
```

Zadatak 3. *(7 bodova)*

a) Zadan je niz brojeva: 10, 15, 2, 17, 3, 40. Ilustrirati (nacrtati stablo nakon svake promjene) stvaranje gomile od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj O(nlogn).

b) Ilustrirati korake uzlaznog *heapsorta* (nacrtati stablo nakon svake zamjene elemenata) na zadanoj gomili.



Zadatak 4. *(7 bodova)*

Neka je zadana objektno orijentirana implementacija stoga sljedećom klasom:

```
typedef int tip;
class Stog{
      struct at {
            tip element;
            struct at *sljed;
      };
      typedef struct at atom;
      atom *vrh;
      void obrisiStog();
public:
      Stog();
      ~Stog();
      int Dodaj (tip element);
      int Skini (tip *element);
};
```

Proširite navedenu implementaciju stoga kako bi bilo moguće realizirati sljedeće podebljano označene retke:

```
Stog stari; int n;
Stog *treci;
…
Stog novi(&stari, n); /*   Stvara novi stog koji sadrži prvih (gornjih) n
                           elemenata starog stoga (ili sve elemente starog
                           stoga ako on sadrži manje od n elemenata), ali
                           u obrnutom poretku. Sadržaj starog stoga mora
                           biti očuvan. */

treci = stari.Skini(n); /* Vraća novi stog koji se sastoji od prvih
                           (gornjih) n elemenata starog stoga (ili od svih
                           elemenata starog stoga ako on sadrži manje od n
                           elemenata) u obrnutom poretku, pritom skidajući
                           te elemente iz starog stoga. */
```

Napomena: prazni konstruktor, destruktor te funkcije `Dodaj` i `Skini` su već napisane i njih ne treba pisati. (*Naravno, smijete ih koristiti!*)

Primjer: Ako je stog stari sadržavao elemente 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 (*vrh je napisan krajnje lijevo*), nakon:

```
Stog novi(&stari, 5);
```

stvara se novi stog koji sadrži elemente 6, 7, 8, 9, 10, dok stari stog ostaje isti.

Nakon:

```
treci = stari.Skini(3);
```

treći stog sadrži elemente 8, 9, 10, a stari stog sadrži 7, 6, 5, 4, 3, 2, 1.

Rješenja

1. zadatak

a)
```
int jeligomila(int stablo[], int n){
      int i;
      for (i = n; i >= 2; i--)
            if (stablo[i] > stablo[i/2])
                  return 0;
      return 1;
   }
```

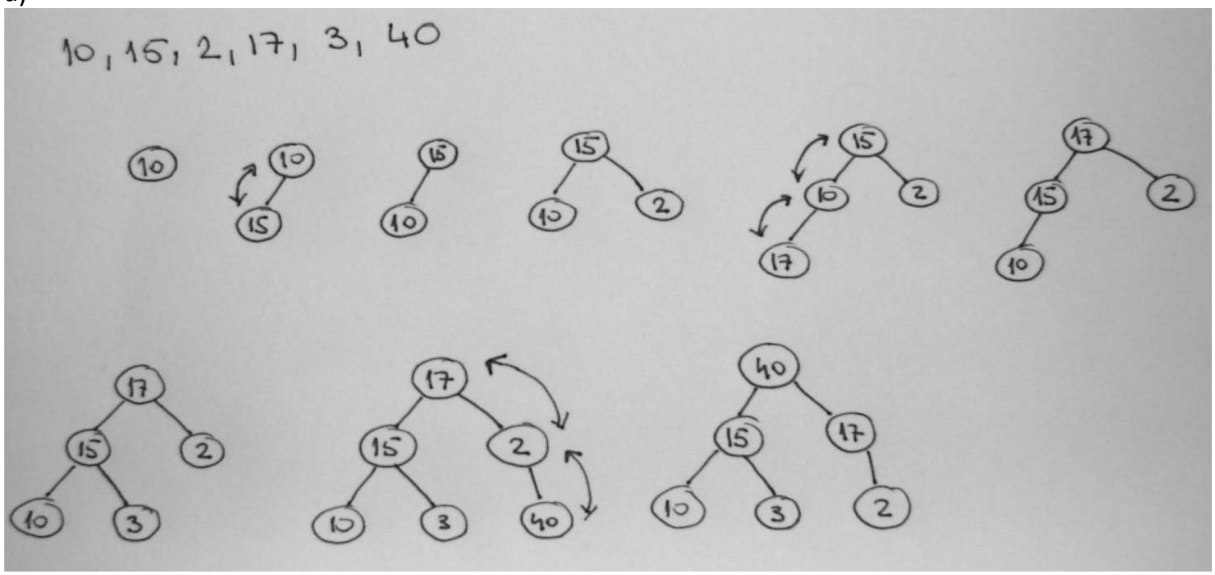
b)
```
int jeligomila(cvor *korijen){
      if (korijen == NULL) return 1;
      if ((korijen->lijevo != NULL)
            && (korijen->lijevo->element > korijen->element)) return 0;
      if ((korijen->desno != NULL)
            && (korijen->desno->element > korijen->element)) return 0;
      return jeligomila(korijen->lijevo) && jeligomila(korijen->desno);
   }
```

2. zadatak

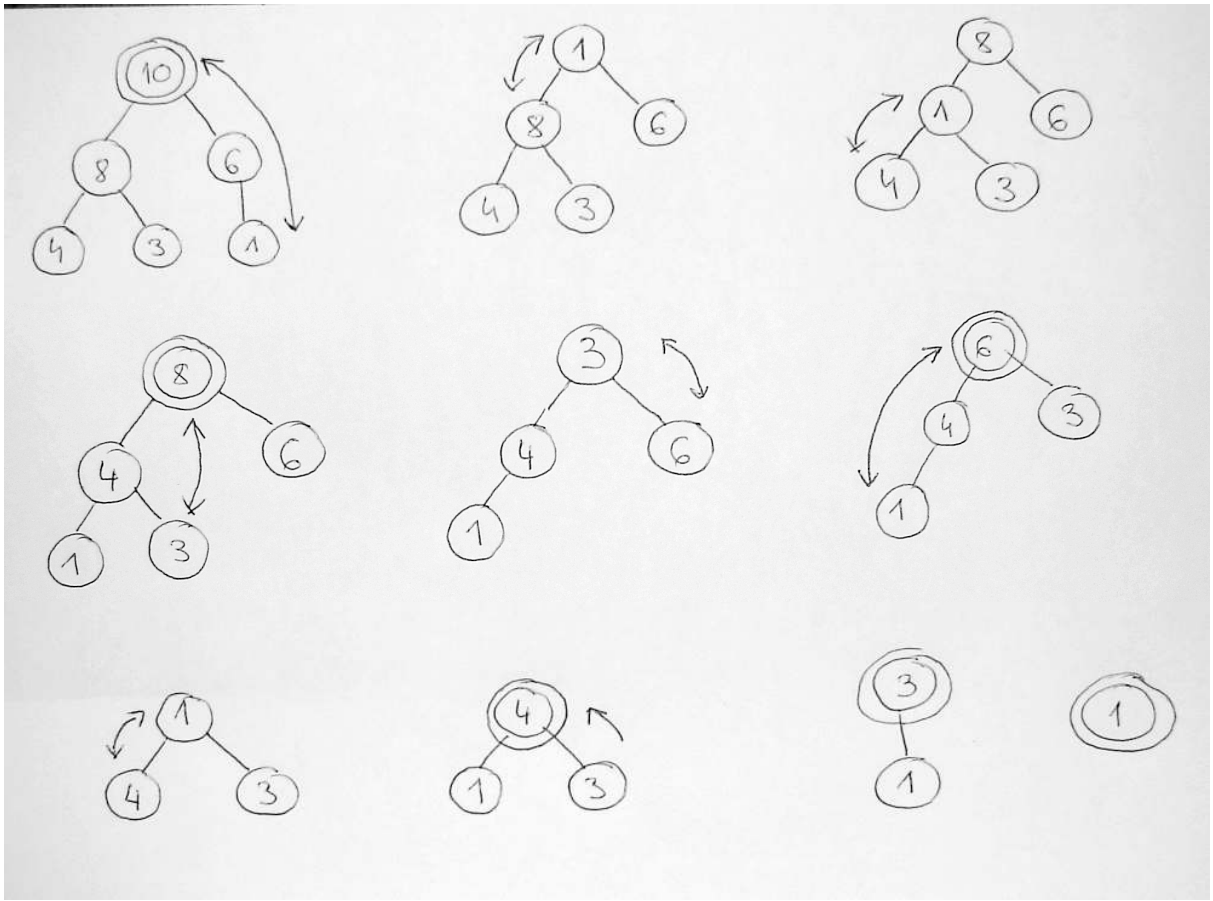
```
void izbaci(atom **glava){
  atom *stari, *prethodni;

  stari = *glava;
  prethodni = NULL;

  while(stari) {
    if (stari->element.godisnji > 30) {

      if (stari == *glava){
        *glava = stari->sljed;
        free(stari);
        stari = *glava;
      } else {
        prethodni->sljed = stari->sljed;
        free(stari);
        stari = prethodni->sljed;
      }

    } else {
      prethodni = stari;
      stari = stari->sljed;
    }
  }
}
```

3. zadatak

a)



b)



4. zadatak

```
Stog::Stog(Stog *stari, int n){
	int i = 1;tip element;Stog pom;
	this->vrh = NULL;
	while(i<=n && stari->Skini(&element)){
		pom.Dodaj(element);
		this->Dodaj(element);
		i++;
	}
	while(pom.Skini(&element)){
		stari->Dodaj(element);
	}
}
Stog* Stog::Skini(int n){
	int i = 1;tip element;
	Stog *novi = new Stog();
	while(i<=n && this->Skini(&element)){
		novi->Dodaj(element);
		i++;
	}
	return novi;
}
```