JMBAG **IME I PREZIME**

Algoritmi i strukture podataka

21. rujna 2016.

Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe goto. Ovaj primjerak ispita trebate predati s upisanim imenom i prezimenom te JMBAG-om.
*Ispit donosi maksimalno 70 bodova, a prag za prolaz pismenog ispita je **35** bodova uz barem jedan točno riješen zadatak.*

Zadatak 1. (15 bodova)

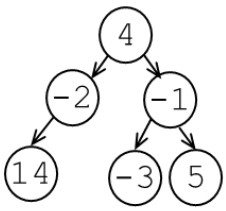
Zadano je binarno stablo čiji su čvorovi definirani strukturom:

```
typedef struct s {
      int element; // vrijednost čvora (cijeli broj različit od 0)
      struct s *lijevo, *desno;
} cvor;
```

Napišite funkciju **najList** koja vraća najveću vrijednost među listovima zadanog binarnog stabla. Funkcija vraća 0, ako je stablo prazno. Prototip funkcije je:

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

Primjer: za stablo na donjoj slici, funkcija **najList** treba vratiti 14.



Zadatak 2. (14 bodova)

Podatke o zračnim lukama potrebno je organizirati u memorijski rezidentnu tablicu raspršenog adresiranja veličine 10000 mjesta. Memorija za tablicu raspršenog adresiranja rezervira se u glavnom programu (nije ga potrebno pisati). Kolizija se rješava *kvadratnim ispitivanjem*. Podrazumijeva se da je mjesto u tablici slobodno, ako je šifra upisanoga zapisa jednaka 0.

a) Potrebno je napisati pretprocesorsku naredbu kojom se definira veličina tablice i ostale potrebne konstante za potrebe kvadratnog ispitivanja ($c_1 = 0$ *i* $c_2 = 1$). Potrebno je definirati i strukturu **zapis** za pohranu podataka o zračnim lukama, koja sadrži sljedeće podatke: šifru (cijeli broj), naziv (najviše 50+1 znak) i oznaku (najviše 5+1 znak).

b) Potrebno je napisati funkciju **prebroji** koja izračunava i vraća broj zapisa u tablici raspršenog adresiranja koji se *ne nalaze* na „pravom“ mjestu, a njihovo je „pravo“ mjesto slobodno. Pod pojmom „pravo“ mjesto podrazumijeva se mjesto u tablici na kojemu bi zapis bio smješten da nema kolizije. Napomena: slobodna mjesta mogu nastati brisanjem elemenata.
Prototip funkcije je:

```
int prebroji(zapis *hash);
```

Na raspolaganju imate unaprijed definiranu funkciju raspršenog adresiranja **adresa** čiji je prototip:

```
int adresa(int sifra);
```

Zadatak 3. (10 bodova)

U polju cijelih brojeva pohranjen je sljedeći niz brojeva:
1, 2, 5, 6, 8, 9, 7, 10
Polje predstavlja **gomilu** s relacijom *manji od* (*min heap*).

Potrebno je **silazno** sortirati zadano polje metodom *heapsort* uz prikazivanje svakog koraka sortiranja (potrebno je napisati sadržaj polja nakon svake promjene).

Zadatak 4. (18 bodova)

Napišite rekurzivnu funkciju **preuredi** koja za zadani pozitivni cijeli broj **n** vraća broj, koji je dobiven tako da se svako pojavljivanje neparnih znamenki (pojava jedne ili više uzastopnih neparnih znamenki) u broju **n** zamijeni točno jednom znamenkom 1.

Poredak parnih znamenaka (uključujući 0) se ne mijenja.

Prototip funkcije je:

```
long preuredi(long n);
```

Primjeri: `preuredi(52331)=121, preuredi(225307)=22101, preuredi(1333)=1`

Napomena: Nerekurzivna rješenja se neće priznavati.

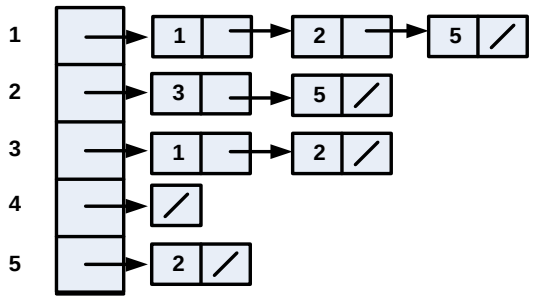
Zadatak 5. (6 bodova)

Zadano je polje od **n** članova tipa `int`. Odredite vrijeme izvođenja u Θ notaciji potrebno za sortiranje polja sljedećim algoritmima posebno razmatrajući **prosječan** (očekivani) te **najgori** slučaj za svaki od algoritama:

	Prosječan (očekivani) slučaj	Najgori slučaj
Sortiranje umetanjem (*insertion sort*)	Θ_____	Θ_____
quicksort	Θ_____	Θ_____

Zadatak 6. (7 bodova)

Za zadanu listu susjedstva, nacrtajte usmjereni graf:



Rješenja:

1. zadatak (15 bodova)

```
int najList(cvor *korijen){
	int l, d;
	if (korijen) return 0; // prazno stablo
	if (!(korijen->lijevo) && !(korijen->desno)) { // ako si list, vrati svoju vrijednost
		return korijen->element;
	}
	if (!(korijen->lijevo)) { // ako nema lijevo dijete, tj. podstablo, gledaj samo desno
		return najList(korijen->desno);
	}
	if (!(korijen->desno)) { // ako nema desno podstablo, gledaj samo lijevo
		return najList(korijen->lijevo);
	}
	// ako ima i lijevo i desno podstablo, pogledaj koji ima najveći list
	l = najList(korijen->lijevo);
	d = najList(korijen->desno);
	if (l > d) return l;
	else return d;
}
```

2. zadatak (14 bodova)

a)

```
#define M 1000
#define c1 0
#define c2 1

typedef struct z {
	int sifra;
	char naziv[50+1];
	char oznaka[5+1];
} zapis;
```

b)

```
int prebroji (zapis *hash) {
	int trazeniBroj = 0;
	int i, prava;

	for (i = 0; i < M; i++) {
		if (hash[i].sifra != 0) { // ako zapis nije prazan
			prava = adresa(hash[i].sifra); // odredi koja je prava adresa
			if (prava != i) {
				// ako zapis nije na pravoj adresi, provjeri je li njegovo „pravo“
				// mjesto slobodno
				if (hash[prava].sifra == 0) { // ako nema ničeg ...
					++ trazeniBroj; // povećaj traženi broj za 1
				}
			}
		}
	}
	return trazeniBroj;
}
```

3. zadatak (10 bodova)

Zadano polje (gomila s relacijom manji od, tj. *minheap*):

1, 2, 5, 6, 8, 9, 7, 10

S**ilazno** sortiranje zadanog polja metodom *heapsort*:

```
10,  2,  5,  6,  8,  9,  7,  1          zamijenjeni su 1 i 10; 10 dođe u korijen
 2,  2,  5,  6,  8,  9,  7,  1          podesi gomilu (bez 1)
 2,  6,  5,  6,  8,  9,  7,  1
 2,  6,  5, 10,  8,  9,  7,  1
 2,  6,  5, 10,  8,  9,  7,  1          zamijeni 2 i 7; 7 dođe u korijen
 7,  6,  5, 10,  8,  9,  2,  1          podesi gomilu (bez 1 i 2)
 5,  6,  5, 10,  8,  9,  2,  1
 5,  6,  7, 10,  8,  9,  2,  1
```

```
 5,  6,  7, 10,  8,  9,  2,  1          zamijeni 5 i 9; 9 dođe u korijen
 9,  6,  7, 10,  8,  5,  2,  1          podesi gomilu
 6,  6,  7, 10,  8,  5,  2,  1
 6,  8,  7, 10,  8,  5,  2,  1
 6,  8,  7, 10,  9,  5,  2,  1
 6,  8,  7, 10,  9,  5,  2,  1          zamijeni 6 i 9; 9 dođe u korijen
 9,  8,  7, 10,  6,  5,  2,  1          podesi gomilu
 7,  8,  7, 10,  6,  5,  2,  1
 7,  8,  9, 10,  6,  5,  2,  1
 7,  8,  9, 10,  6,  5,  2,  1          zamijeni 7 i 10; 10 dođe u korijen
10,  8,  9,  7,  6,  5,  2,  1          podesi gomilu
 8,  8,  9,  7,  6,  5,  2,  1
 8, 10,  9,  7,  6,  5,  2,  1
 8, 10,  9,  7,  6,  5,  2,  1          zamijeni 8 i 9; 9 dođe u korijen
 9, 10,  8,  7,  6,  5,  2,  1          podesi gomilu
10,  9,  8,  7,  6,  5,  2,  1          zamijeni 9 i 10 i podesi gomilu
10,  9,  8,  7,  6,  5,  2,  1          sortirano polje
```

4. zadatak (18 bodova)

```
long preuredi(long n){
	int m;
	long p;

	if (n < 10) { // ovo je kraj rekurzivnih poziva ...
		if (n % 2 == 0) return n;
		else return 1;
	}
	else p = preuredi(n / 10);

	m = n % 10; // znamenka jedinica ulaza...
	if (m % 2 == 0) { // ako je m paran, dodaj ga na kraj
		return p * 10 + m;
	}
	else { // ako je m neparan, pogledaj zadnju znamenku od p,
		if (p % 10 == 1) { // ako je to znamenka 1, ne radi ništa
			return p;
		}
		else { // ako nije, stavi 1 na kraj
			return p * 10 + 1;
		}
	}
}
```

5. zadatak (6 bodova)

	Prosječan (očekivani) slučaj	Najgori slučaj
Sortiranje umetanjem (*insertion sort*)	$\Theta(n^2)$	$\Theta(n^2)$
Quicksort	$\Theta(n \log n)$	$\Theta(n^2)$

6. zadatak (7 bodova)