ISPIT IZ ALGORITAMA I STRUKTURA PODATAKA *4. jul 2007.*

1. [20] Napisati potprograme na programskom jeziku C ili C++ za rad sa heš tabelom, primenom metode objedinjenog ulančavanja. Ključevi su nenegativni celi brojevi. Obezbediti sledeće funkcionalnosti: stvaranje tabele sa zadatim brojem ulaza, umetanje ključa, dohvatanje ključa (vraća negativnu vrednost ako ključ ne postoji u tabeli). Napisati glavni program koji demonstrira korišćenje navedenih potprograma.

2. [14] Dekodirati LZW algoritmom sledeću poruku: 1 0 2 3 5 6 2 4. Osnovna tabela simbola sadrži dva simbola: simbolu A odgovara kod 0, a simbolu B kod 1.

3. [16] U prazno B-stablo reda 3 umeću se redom ključevi 22, 16, 30, 40, 8, 4, 10, 20, 12, 18, 15 a zatim se redom brišu ključevi 15, 22, 18, 10. Nacrtati izgled stabla nakon svake od navedenih izmena. Koliki je srednji broj pristupa prilikom uspešnog i neuspešnog traženja, kao i popunjenost B stabla, posle svih umetanja ključeva i u završnom stanju?

5. [18] Pitanja:
   a) Izvesti adresnu funkciju za smeštanje matrice $X[l1{:}l2,\ u1{:}u2]$ po kolonama.
   b) Objasniti kada i zašto u kružnoj ulančanoj listi ima smisla da spoljašnji pokazivač pokazuje na poslednji čvor u listi.
   c) Objasniti algoritam za generisanje koda za nulaadresnu mašinu iz izraza u postfiksnoj notaciji. Dati primer.
   d) Uporediti realizacije reda za čekanje sa nizovima $Q[1{:}n]$ i $Q[0{:}n-1]$.

5. [16] Skicirati i objasniti algoritam za generisanje dubinskog obuhvatnog stabla. Razmotriti i iterativnu realizaciju. Izvesti složenost za obe reprezentacije grafa..

6. [16] Objasniti algoritam sortiranja direktnim umetanjem u nerastućem poretku i ilustrovati ga na primeru niza 27 14 18 30 6 17 11. Izračunati broj poređenja i premeštanja u najboljem, najgorem i srednjem slučaju. Kako se broj poređenja i premeštanja može smanjiti i kako to utiče na složenost?

Ispit traje 4 h

Rešenja:

1. Rešenje je prikazano podebljanim slovima.

```
// HashTable.h
#include <iostream>
using namespace std;

class HashTable
{
protected:
  int velicina;
  int slobodna;

  struct Element
  {
     int kljuc;
     int sledeci;
     Element():kljuc(-1), sledeci(-1) { }
  };

  Element *tabela;

  int pronadji_ulaz(int kljuc) const;
  int h(int kljuc) const
  { return kljuc % velicina; }

public:
   HashTable(int vel);
   virtual ~HashTable() { if( tabela ) delete []tabela; }y

  int umetni(int kljuc);
  int dohvati(int kljuc);

  friend ostream & operator<<(ostream &os,
                              const HashTable &table);
};
```

```
// HashTable.cpp
#include "HashTable.h"
HashTable::HashTable(int vel) {
      velicina = vel;
      slobodna = vel-1;
      tabela = new Element[vel];
}

unsigned int HashTable::pronadji_ulaz(unsigned int kljuc) const {
      unsigned int i = h(kljuc);
      while( tabela[i].kljuc != kljuc && tabela[i].sledeci != -1)
            i = tabela[i].sledeci;
      return i;
}
```

```
int HashTable::umetni(int kljuc) {
      int ind = pronadji_ulaz(kljuc);
      if( tabela[ind].kljuc == kljuc ) return 0;
      int j;
      if( tabela[ind].kljuc == -1 ) j = ind;
      else  {
            while( tabela[slobodna].kljuc != -1 )
               if( --slobodna < 0 ) throw "Tabela je prepunjena";
            j = slobodna; tabela[ind].sledeci = slobodna;
      }
      tabela[j].kljuc = kljuc;
      return 1;
}

int HashTable::dohvati(unsigned int kljuc) {
      unsigned int i = pronadji_ulaz(kljuc);
      if( tabela[i].kljuc == kljuc ) return i;
      return -1;
}

ostream & operator<<(ostream &os, const HashTable &tabela) {
      for(int i = 0; i < tabela.velicina; i++)       {
         os << i << '\t' << tabela.tabela[i].kljuc << '\t' <<
                              tabela.tabela[i].sledeci;
         if( tabela.slobodna == i ) os << '\t' << "slob";
         os << endl;
      }
      return os;
}
```

```
// Glavni program
#include<iostream>
using namespace std;
#include "HashTable.h"
void main() {
HashTable tabela(20);
      try {
         for(int i = 5; i < 120; i+= 6)  tabela.umetni(i);
         cout << tabela;
         if( tabela.dohvati(10) == -1 ) {
           cout << "Kljuc 10 ne postoji" << endl;
           tabela.umetni(10);
         }
         if( tabela.umetni(10)==0) cout<<"Kljuc 10 vec postoji"<<endl;
         tabela.umetni(11); tabela.umetni(12);
      }
      catch(char *s) { cout << s << endl;  }
      cout << tabela;
}
```

2.

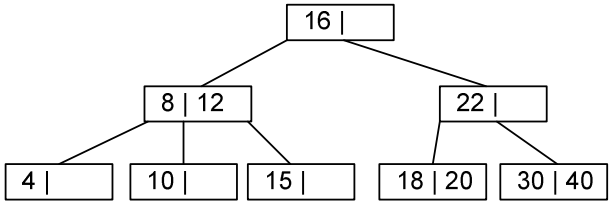
Dekodovana poruka: BABAABABAABAABABAA

OLD_CODE	NEW_CODE	STRING	CHARACTER	TABLE		OUTPUT
1						B
1	0	A	A	BA	2	A
0	2	BA	B	AB	3	BA
2	3	AB	A	BAA	4	AB
3	5	ABA	A	ABA	5	ABA
5	6	ABAA	A	ABAA	6	ABAA
6	2	BA	B	ABAAB	7	BA
2	4	BAA	B	BAB	8	BAA

Tabela simbola nakon dekodiranja poruke:

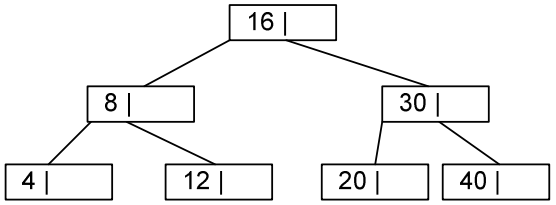
A	0
B	1
BA	2
AB	3
BAA	4
ABA	5
ABAA	6
ABAAB	7
BAB	8

3.



Prosečan broj pristupa:
Uspešno: 28/11 = 2.55
Neuspešno: 3

Popunjenost: 11/16 = 0.69



Prosečan broj pristupa:
Uspešno: 17/7 = 2.43
Neuspešno: 3

Popunjenost: 7/14 = 0.5