**JMBAG** **IME I PREZIME**

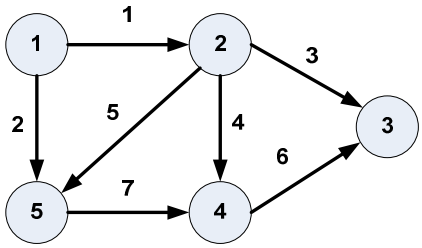
# Algoritmi i strukture podataka – završni ispit

*21. lipnja 2016.*

*Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe goto. Ispit donosi maksimalno 40 bodova. Ovaj primjerak ispita trebate predati s upisanim imenom i prezimenom te JMBAG-om.*

## *Zadatak 1. (6 bodova)*

Za zadani graf nacrtajte matrice susjedstva i incidencije te listu susjedstva.




## *Zadatak 2. (10 bodova)*

Potrebno je sortirati zadano polje cijelih brojeva na način da ga se prvo prepiše u binarno stablo za pretraživanje, a zatim ga se, inorder postupkom, prepiše natrag u polje. Možete pretpostaviti da već postoji funkcija

```
cvor *dodaj(int br, cvor *korijen)
```

koja zadani element dodaje u binarno stablo te vraća pokazivač na novonastalo stablo. Čvorovi stabla imaju sljedeću strukturu:

```
typedef struct cv{
      int podatak;
      struct cvor *lijevo_dijete;
      struct cvor *desno_dijete;
} cvor;
```

a) Napisati funkciju koja će elemente zadanog binarnog stabla za pretraživanje prepisati u jednodimenzionalno polje, tako da to polje bude sortirano uzlazno. Primjer prototipa funkcije:

```
void prepisi(cvor *korijen, int *i, int *polje);
```

Varijabla *i* određuje na kojem mjestu u polju počinje zapisivanje elemenata stabla. Funkcija treba osloboditi svu memoriju koju su elementi stabla zauzeli.

b) Napisati funkciju koja će sortirati zadano polje cijelih brojeva pomoću binarnog stabla za pretraživanje.

Funkcija treba imati prototip:

```
void mojsort(int *polje, int n);
```

## *Zadatak 3. (8 bodova)*

Napišite funkciju koja će za zadano polje prirodnih brojeva provjeriti je li gomila. U polju su s -1 označeni elementi koji se ne koriste. Funkcija treba vratiti 1 ako je zadano polje gomila, a -1 ako nije. Funkcija treba imati prototip:

```
int jeliGomila(int *polje, int n, int vrstaGomile);
```

Pomoću varijable `vrstaGomile`, određuje se za koju vrstu gomile se obavlja ispitivanje. Ako je `vrstaGomile = 1` roditelj treba biti veći od djece, inače roditelj treba biti manji od djece. Možete pretpostaviti da gomila kreće od indeksa 1.

*Zadatak 4. (8 bodova)*

Podatke o djelatnicima potrebno je organizirati u memorijski rezidentnu tablicu raspršenog adresiranja veličine 5000 zapisa. Memorija za tablicu raspršenog adresiranja rezervira se u glavnom programu (nije ga potrebno pisati). Kolizija se rješava dvostrukim raspršenim adresiranjem.

Potrebno je napisati:

a) Pretprocesorsku direktivu kojom se definira veličina tablice i ostale potrebne konstante za potrebe dvostrukog raspršenog adresiranja. Potrebno je napisati i strukturu za pohranu podataka o djelatniku. Svaki djelatnik ima šifru (cijeli broj), ime (maksimalna veličina 20 znakova) i prezime (maksimalna veličina 30 znakova).
b) Funkciju za upis u tablicu raspršenog adresiranja koja pojavu kolizija rješava dvostrukim raspršenim adresiranjem. Funkcija vraća 1 ako je upis uspio, a 0 ako je hash-tablica puna. Prototip funkcije je:
`int Upis(struct zapis* hash, struct zapis element);`

Prilikom realizacije funkcije za upis koristiti predefinirane hash-funkcije:

```
int Adresa1(int sifra) {
	return sifra%M;
}

int Adresa2(int sifra) {
	return 1 + sifra%(M - 1);
}
```

*Zadatak 5. (8 bodova)*

Za tip podatka `Stog` definirane su funkcije za inicijalizaciju stoga, dodavanje elementa na stog i skidanje elementa sa stoga:

```
void init_stog(Stog *stog);
int dodaj(Stog *stog, int element);
int skini(Stog *stog, int* element);
```

Napisati funkciju

```
void izbaci(Stog *stog);
```

koja će izbaciti sa stoga sve vrijednosti koje se na stogu pojavljuju više od jedanput, pri čemu će u slučaju višestrukog pojavljivanja vrijednosti biti zadržana vrijednost koja se pojavljuje prvi put promatrano od vrha stoga.

**Primjer**:

Stog prije ulaska u fukciju:

| |
|---|
| 1 |
| 1 |
| 4 |
| 3 |
| 2 |
| 7 |
| 2 |
| 3 |
| 1 |

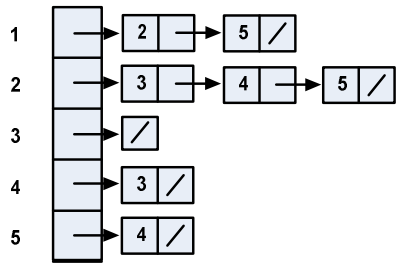
Stog nakon izlaska iz fukcije:

| |
|---|
| 1 |
| 4 |
| 3 |
| 2 |
| 7 |

**Napomena**: Rješenje mora koristiti isključivo funkcije za rad sa stogom i pomoćne stogove. Sva rješenja koja podrazumijevaju implementacijske detalje stoga (realizacija poljem ili listom) neće biti priznata.

**Rješenja:**

1. **Zadatak**




$$Adj = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad Inc = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & -1 \\ 0 & -1 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

2. **Zadatak**

```
// Funkcija koja prebacuje elemente stabla inorder u polje,
// a ujedno i oslobađa memoriju
void prepisi (cvor *korijen, int *i, int *polje) {
  if (korijen == NULL) {
    return;
  }

  prepisi (korijen->lijevo_dijete, i, polje);
  polje[*i] = korijen->podatak;
  (*i)++;
  prepisi (korijen->desno_dijete, i, polje);

  free(korijen);           // Oslobađanje memorije
}

// Funkcija koja sortira zadano polje cijelih brojeva tako da ga prepiše
// u binarno stablo za pretraživanje, a zatim ga inorder prepiše natrag u polje
void mojsort(int *polje, int n) {
  int i;
  cvor *korijen;

  korijen = NULL;
  for (i=0; i<n; i++) {
    korijen = dodaj(polje[i], korijen);
  }
  i = 0;
  prepisi (korijen, &i, polje);
}
```

3. **Zadatak**

```
int jeliGomila(int *polje, int n, int vrstaGomile) {
  int dijete, roditelj;
  int i;

  // Je li riječ o potpunom stablu
  for (i=1; i<=n; i++) {
        if (polje[i] < 0) return -1;
  }

  // Odnos roditelja i djece
  for (dijete=n; dijete>1; dijete--) {
    roditelj = dijete/2;
    if (vrstaGomile == 1 && polje[dijete] > polje[roditelj]) return -1;
    if (vrstaGomile != 1 && polje[dijete] < polje[roditelj]) return -1;
  }
  return 1;
}
```

## 4. Zadatak

a)

```
#define VELIME 20
#define VELPREZIME 30
#define M 5000

struct zapis{
	int sifra;
	char ime[VELIME + 1];
	char prezime[VELPREZIME + 1];
};
```

b)

```
int Upis(struct zapis* hash, struct zapis element) {
	int indeks;
	int h1 = Adresa1(element.sifra);
	int h2 = Adresa2(element.sifra);
	for (int i = 0; i < M; i++) {
		indeks = (h1 +i*h2)%M;
		if (hash[indeks].sifra == 0) {
			hash[indeks] = element;
			return 1;
		}
	}
	return 0;
}
```

## 5. Zadatak

```
void izbaci(Stog *stog) {
Stog s1, s2;
int el, x, nadjen;

	init_stog(&s1);
	init_stog(&s2);
	while(skini(stog, &el)) {
		nadjen = 0;
		while(skini(&s1, &x))
		{
			dodaj(&s2, x);
			if (x == el) nadjen = 1;
		}
		if (!nadjen) dodaj(&s2, el);
		while(skini(&s2, &x)) dodaj(&s1, x);
	}
	while(skini(&s1, &x)) dodaj(&s2, x);
	while(skini(&s2, &x)) dodaj(stog, x);
}
```

ILI

```
void izbaci(Stog *stog) {
Stog s1, s2;
int el, x;

	init_stog(&s1);
	init_stog(&s2);
	while(skini(stog, &el)) {
		dodaj(&s1, el);
		while(skini(stog, &x)) if (x != el) dodaj(&s2, x);
		while(skini(&s2, &x)) dodaj(stog, x);
	}
	while(skini(&s1, &el)) dodaj(stog, el);
}
```