JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – međuispit

19. travnja 2017.

Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe goto. Ispit donosi maksimalno 30 bodova. Ovaj primjerak ispita trebate predati s upisanim imenom i prezimenom te JMBAG-om.

Zadatak 1. (7 bodova)

a) Napišite rekurzivnu funkciju koja za znakovni niz (*string*) proizvoljne duljine provjerava ima li svaka otvorena uglata zagrada u nizu (znak '[') pripadajuću zatvorenu uglatu zagradu (znak ']') i obratno. Funkcija treba vratiti 1, ako svaka otvorena zagrada u nizu ima pripadajuću zatvorenu zagradu, a 0 inače.
Pretpostavite da se znakovni niz sastoji samo od znakova '[' i ']'. Prototip funkcije je:

```
int provjeriZagrade(char *niz, int *zastavica);
```

Naputak: `zastavica` je pomoćna vrijednost koja se koristi za praćenje broja neuparenih zagrada.

Primjeri:
Za niz "**[**[[][]]" funkcija treba vratiti 0, jer zagrada **[** nema pripadajuću zatvorenu zagradu.
Za niz "]][[[]**]**" funkcija treba vratiti 0, jer zagrada **]** nema pripadajuću otvorenu zagradu.
Za niz "[[[][]]]" funkcija treba vratiti 1, jer svaka otvorena zagrada ima pripadajuću zatvorenu zagradu.
Napomena: nerekurzivno rješenje se neće priznati.

b) Napišite odsječak glavnog programa u kojem se poziva funkcija iz a) dijela zadatka za niz "[[]".

Komentar uz zadatak: uparenost zagrada znači da svaka otvorena uglata zagrada (znak '[') ima pripadajuću zatvorenu uglatu zagradu (znak ']'), gdje nije bitno da je redoslijed zagrada matematički ispravan, tj. i niz " []" i niz "][" su ispravno zadani.

Zadatak 2. (7 bodova)

Odredite vrijeme izvođenja u O, Ω i, gdje je moguće, Θ notaciji za funkcije **f1** i **f2**. Ako se vrijeme izvođenja u Θ notaciji ne može odrediti, navedite tako u rješenju. Rješenja upišite u tablice pored zadataka.

a)

```
/* A je polje n cijelih brojeva.
 * Funkcije za sortiranje sortiraju niz uzlazno,
 * a koriste algoritam naveden u imenu funkcije.
*/
void f1(int *A, int n) {
  insertionSort(A, n);
  mergeSort(A, n);
}
```

O	
Ω	
Θ	

b)

```
void f2(int n) {
  int i;
  if (n == 1) {
    return;
  }
  for (i = 0; i < n; i++) {
    f2(n - 1);
  }
}
```

O	
Ω	
Θ	

Okrenite stranicu

Zadatak 3. (8 bodova)

U polje cijelih brojeva pohranjen je sljedeći niz brojeva: **`2, 1, 4, 5, 9, 3, 6, 7, 8.`** Ilustrirati uzlazno sortiranje algoritmom **Quicksort**. Stožer za quicksort odabrati metodom aproksimacije medijana temeljem prvog, središnjeg i zadnjeg člana. Polja s manje ili točno `cutoff = 3` elementa sortirati *insertion sortom*. Navesti korake u sortiranju kod kojih je došlo do zamjene pojedinih članova polja te označiti zamjene članova polja. Također, detaljno prikazati korake zamjene kod sortiranja *insertion sortom*.

Zadatak 4. (8 bodova)

U jednostruko povezanu nesortiranu listu spremljeni su cijeli brojevi. Lista je zadana sljedećom strukturom:

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

Napišite funkciju prototipa:

```
void kopirajListu(atom* glava_ulaz, atom** glava_izlaz);
```

koja će elemente liste zadane pokazivačem `glava_ulaz` kopirati u novu listu zadanu pokazivačem `glava_izlaz`. Zapisi u izlaznoj listi moraju biti u istom poretku kao i zapisi u ulaznoj listi. Prilikom kopiranja elemenata u izlaznu listu potrebno je rezervirati memoriju za elemente, a ulaznu listu ostaviti nepromijenjenom.

Rješenja:

1. zadatak (7 bodova)

```
#include <stdio.h>
int provjeriZagrade(char *niz, int *zastavica) {
  if (*niz == '\0') {
    return (*zastavica == 0 ? 1 : 0);
  }
  else {
    if (*niz == '[') ++(*zastavica);
    else --(*zastavica);
    return provjeriZagrade(niz + 1, zastavica);
  }
}

int main() {
  int zastavica = 0;
  if (provjeriZagrade("[[]", &zastavica) == 1) {
    printf("Sve zagrade su uparene");
  }
  else {
    printf("Neke zagrade nisu uparene");
  }
}
```

2. zadatak (7 bodova)

a) $O(n^2)$, $\Omega(n \log n)$ (najbolji slučaj: polje je već uzlazno sortirano; najlošiji slučaj: silazno sort. polje)

b) $O(n!)$, $\Omega(n!)$, $\Theta(n!)$

3. zadatak (8 bodova)

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

4. zadatak (8 bodova)

```
void kopirajListu(atom* glava_ulaz, atom** glava_izlaz){
	atom *pom;
	while (glava_ulaz != NULL){
		if (*glava_izlaz == NULL)
		{
			*glava_izlaz = malloc(sizeof(atom));
			(*glava_izlaz)->element = glava_ulaz->element;
			(*glava_izlaz)->sljed = NULL;
			pom = *glava_izlaz;
		}
		else{
			pom->sljed = malloc(sizeof(atom));
			pom->sljed->element = glava_ulaz->element;
			pom->sljed->sljed = NULL;
			pom = pom->sljed;
		}
		glava_ulaz = glava_ulaz->sljed;
	}
}
```

Alternativno rješenje (korištenje adrese pokazivača sljed):

```
void kopirajListu2(atom* glava_ulaz, atom** glava_izlaz){
	while (glava_ulaz != NULL){
		*glava_izlaz = malloc(sizeof(atom));
		(*glava_izlaz)->element = glava_ulaz->element;
		(*glava_izlaz)->sljed = NULL;
		glava_ulaz = glava_ulaz->sljed;
		glava_izlaz = &((*glava_izlaz)->sljed);
	}
}
```