REDOVI

2013. zavrsni

Zadatak 1. (6 bodova)

Za tip podatka Red definirane su funkcije (nije ih potrebno pisati) za inicijalizaciju reda, dodavanje elementa u red te za skidanje elementa iz reda. Prototipovi navedenih funkcija su:

```
void init_red(Red *red);
int dodaj(Element element, Red *red);
int skini(Element *element, Red *red);
```

Funkcije **dodaj** i **skini** vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače. Elementi reda su podaci tipa **Element**. Tipovi **Element** i **Red** definirani su redom sljedećim programskim odsječkom:

```
#define MAXNIZ 20
typedef struct Element{
    char niz[MAXNIZ + 1];
} Element;
```

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

```
typedef struct {
      atom *ulaz;
      atom *izlaz;
} Red;
```

Napišite funkciju koja će iz zadanog reda u novi red kopirati sve elemente čija varijabla `niz` sadrži podniz koji funkcija primi kao ulazni argument. Funkcija pozivatelju vraća taj novi red, a prototip joj je:

```
Red* trazi(Red *red, char* podniz)
```

Poredak elemenata u novom redu mora biti isti kao i poredak u ulaznom redu. Po završetku izvođenja funkcije, ulazni red mora sadržavati iste podatke u istom redoslijedu kao i prije poziva funkcije `trazi`.

Ako npr. zadani red sadrži elemente sa sljedećim nizovima znakova: "234dd6", "aBc569", "1234d", "ab45de", a traženi podniz je "34d", funkcija mora vratiti novi red čiji elementi sadrže nizove "234dd6" i "1234d".

Napomene: Rješenja koja neće koristiti zadane funkcije za rad s redom donijet će 0 bodova. Možete koristiti pomoćne redove unutar funkcije **trazi**. *Skrećemo pozornost da bi funkcije `strcpy` i `strstr` iz standardne programske knjižnice C-a mogle biti od koristi.*

Zadatak 1. (6 bodova)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

Red* trazi(Red *red, char* podniz){
    Red *rezultat = NULL, pom;
    Element element;
    /*Inicijaliziraj redove*/
    rezultat = (Red*)malloc(sizeof(Red));
    init_red(rezultat);
    init_red(&pom);
    /*Iteracija po svim elementima reda*/
    while(skini(&element, red)){
        /*Provjeri sadrzi li element reda zadani podniz*/
        if(strstr(element.niz, podniz)){
            /*Dodaj novi element u red koji moramo vratiti.
           Ovdje se pretpostavilo da 'dodaj' radi kopiranje.
           Rješenje koje to ne pretpostavlja je isto u redu*/
           dodaj(element, rezultat);
        }
        /*Dodaj element na pomocni red*/
        dodaj(element, &pom);
    }
    /*Vrati elemente na originalni red*/
    while(skini(&element, &pom)){
        dodaj(element, red);
    }
    return rezultat;
}
```

2012.zavrsni

Zadatak 1. (6 bodova)

Za red realiziran jednostruko povezanom listom napišite funkciju **dodaj** za dodavanje elementa u red. Prototip funkcije je:

```
int dodaj(int element, atom **ulaz, atom **izlaz);
```

Funkcija treba vratiti 1 ukoliko je element uspješno dodan u red, a 0 inače.
Tip `atom` definiran je sljedećim programskim odsječkom:

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

Zadatak 1. (6 bodova)

```
int dodaj(int element, atom **ulaz, atom **izlaz){
  atom *novi;
  if (novi = malloc (sizeof (atom))) {
    novi->element = element;
    novi->sljed = NULL;
    if (*izlaz == NULL)
      *izlaz = novi;  // ako je red bio prazan
    else
      (*ulaz)->sljed = novi; // inace, stavi na kraj
    *ulaz = novi;         // zapamti zadnjeg
    return 1;
  }
  return 0;
}
```

Zadatak 2. (13 bodova)

Za tip podatka Red definirane su funkcije (nije ih potrebno pisati) za inicijalizaciju reda, dodavanje elementa u red te za skidanje elementa iz reda. Prototipovi navedenih funkcija su:

```
void init_red(Red *red);
int dodaj(int element, Red *red);
int skini(int *element, Red *red);
```

Funkcije **dodaj** i **skini** vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače. Elementi reda su podatci tipa **int.** Tip **Red** definiran je sljedećim programskim odsječkom:

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

typedef struct {
	atom *ulaz;
	atom *izlaz;
} Red;
```

Napišite funkciju koja će iz zadanog reda u novi red izdvojiti sve elemente koji su manji od prosjeka elemenata u redu. U ulaznom redu trebaju ostati elementi koji nisu izdvojeni. Funkcija pozivatelju vraća taj novi red, a prototip je:

Red* IzdvojiManje(Red *red)

Zad 2.

```
Red *IzdvojiManje(Red *red){
		Red pom; Red *novi;
		int br=0,el; double suma=0;
		novi = (Red*)malloc(sizeof(Red));
		init_red(&pom);init_red(novi);
		while(SkiniIzReda(&el, red))
		{
			suma+=el;
			br++;
			DodajURed(el, &pom);
		}
		suma=suma/br;
		while (SkiniIzReda(&el,&pom))
		{
			if(el<suma) DodajURed(el,novi);
			else DodajURed(el,red);
		}
		return novi;
}
```

Ljetni 2012

5. zadatak (11 bodova)

Za tip podatka Red koji je realiziran jednostruko povezanom listom definirane su funkcije za inicijalizaciju reda, dodavanje elementa u red i skidanje elementa iz reda:

```
void init_red(Red *red);
int dodaj (Kupac element, Red *red);
int skini (Kupac *element, Red *red);
```

Funkcije `dodaj` i `skini` vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače.
Elementi reda su podaci tipa `Kupac` (šifra kupca (cijeli broj) i godina rođenja (cijeli broj)).

Korištenjem gore navedenih funkcija napišite funkciju **`najstarijiNaprijed`** koja će na početak reda staviti najstariju osobu u redu (prema godini rođenja). Ako takvih osoba ima više od jedne, one sve trebaju doći na početak reda u istom poretku u kojem su bile prije promjene. Poredak ostalih osoba u redu mora ostati nepromijenjen. Smiju se koristiti samo zadane funkcije i eventualno pomoćni redovi. Dan je primjer:

Prije promjene: izlaz->12;1988->75;**1945**->13;1946->87;1968->97,1968->23;**1945**<-ulaz
Nakon promjene: izlaz->75;**1945**->23;**1945**->12;1988->13;1946->87;1968->97,1968<-ulaz

5. (11 bodova)

```
void starijiNaprijed(Red *red){
  Red pom1, pom2;
  Kupac k;
  int najstariji, prvi=1;
  init_red (&pom1);
  init_red (&pom2);
  while (skini (&k, red)){
    if (prvi) {
      prvi=0;
      najstariji=k.godinarodj;
    }
    else if (k.godinarodj <najstariji) najstariji=k.godinarodj;
    dodaj (k,&pom1);
    dodaj (k,&pom2);
  }
  while (skini (&k, &pom1)){
    if (k.godinarodj == najstariji) dodaj (k,red);
  }
  while (skini (&k, &pom2)){
    if (k.godinarodj !=najstariji) dodaj (k,red);
  }
}
```