Stog

Zavrsni 2014.

Zadatak 2. (6 bodova)

Definiran je tip podataka (varijabli) `Red` koji odgovara strukturi red čiji elementi su riječi (nizovi znakova). Za dohvat sljedećeg elementa reda koristi se funkcija:

```
char *redSkini(Red *red);
```

Također je definirana i struktura `Stog` koja se koristi za LIFO pohranu spomenutih redova, a za rad sa stogom na raspolaganju su funkcije:

```
void stogInit(Stog *stog);
void stogDodaj(Stog *stog, Red *element);
Red *stogSkini(Stog *stog);
```

Funkcija za dohvaćanje (skidanje) sa stoga vraća NULL ako je stog prazan.

Važno! Redovi riječi imaju strogo propisanu strukturu kao na slici, dakle svaki red osim posljednjeg (prvog stavljenog na stog) sadrži točno jednu riječ „PREKID“, dok posljednji red ne sadrži tu riječ. Također, može se pretpostaviti da stog nije prazan, tj. barem je jedan red riječi na stogu.

Potrebno je napisati funkciju `ispis` prototipa

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

koja ispisuje sve riječi svih redova na stogu na standardni izlaz tako da se trenutačni red ispisuje riječ po riječ sve do riječi „PREKID“. Riječ „PREKID“ se ne ispisuje, nego se nastavlja ispisivanjem sadržaja sljedećeg reda na stogu. Nakon što je sljedeći red na stogu ispisan, funkcija nastavlja ispisivati ostatak sadržaja prethodnog reda (čiji je ispis bio prekinut). Za stog i redove u primjeru na slici, ispis treba biti `12<abcd>34`.

Nakon povratka iz funkcije `ispis`, stog mora ostati sačuvan (iako redovi čije su adrese na stogu ne moraju).

Zadatak 2. (6 bodova)

```
void ispisi( Stog *stog ) {
	Red *pomRed;
	char *rijec;
	Stog ostaci;
	stogInit( &ostaci );

	pomRed = stogSkini( stog );
	while( pomRed != NULL ) {
		rijec = redSkini( pomRed );
		if( rijec == NULL ) {
			stogDodaj( stog, pomRed );
			pomRed = stogSkini( &ostaci );
		}
		else if( strcmp( rijec, "PREKID" ) == 0 ) {
			stogDodaj( &ostaci, pomRed );
			pomRed = stogSkini( stog );
		}
		else {
			printf( "%s", rijec );
		}
	}
}
```

Zadatak 5. (6 bodova)

Zadan je tip podatka **Stog** za koji su definirane funkcije za inicijalizaciju stoga, dodavanje elementa na stog i brisanje elementa sa stoga. Elementi stoga su podaci **tipa Student** koji sadrže prezime studenta (40 znakova), ime studenta (30 znakova) te broj bodova ostvarenih na predmetu ASP. Prototipovi navedenih funkcija su:

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

Funkcije `dodaj` i `skini` vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače.

a) **(1 bod)** Napišite **definicije** odgovarajućih struktura (tipovi podataka **Student, Stog** i **atom**) za stog realiziran vezanom listom.

b) **(5 bodova)** Napišite funkciju koja će sa stoga ukloniti sve studente koji imaju broj bodova manji od prosjeka bodova svih studenata na stogu. Redoslijed studenata koji su ostavljeni na stogu mora biti isti kao i prije poziva funkcije. Funkcija vraća broj studenata uklonjenih sa stoga. Prototip funkcije treba biti:

```
int MakniIspodProsjeka (Stog *stog);
```

Napomene: Rješenja koja neće koristiti zadane funkcije za rad s stogom donijet će 0 bodova. Možete koristiti pomoćne stogove unutar funkcije **MakniIspodProsjeka**.

Zadatak 5. (bodova)

```
typedef struct
{
    char Prezime[40+1];
    char Ime[30+1];
    int Bodova;
} Student;

struct at {
    Student element;
    struct at *sljed;
};

typedef struct at atom;

typedef struct{
    atom *vrh;
} Stog;

int MakniIspodProsjeka(Stog *stog){
    Stog pom;
    Student el;
    int br=0; float sum = 0;float pros;
    init_stog(&pom);
    while (skini(&el,stog)){
        sum+=el.Bodova;
        br++;
        dodaj(el, &pom);
    }
    pros=sum/br;
    br = 0;
    while (skini(&el,&pom)){
        if(pros>=el.Bodova){
            dodaj(el, stog);
        }else{
            br++;
        }
    }
    return br;
}
```

Zadatak 4. (14 bodova)

Za tip podatka Stog koji je realiziran jednostruko povezanom listom definirane su funkcije za inicijalizaciju stoga, dodavanje elementa na stog te skidanje elementa sa stoga. Elementi stoga su nizovi znakova koji sadrže po jedan operator ("+" ili "*") ili cijeli broj zapisan kao niz znakova. Prototipovi navedenih funkcija su:

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

Funkcije `dodaj` i `skini` vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače.

Napisati **rekurzivnu** funkciju `void izracunaj(Stog *stog)` koja će izračunati matematičke izraze pohranjene na stogu u notaciji u kojoj svaki operator slijedi nakon svih svojih operanada. Pretpostavlja se da su operatori binarni, dakle djeluju na po dva operanda (cijeli broj ili izraz kojemu je brojčana vrijednost cijeli broj).

Rezultat izraza nakon izvršavanja funkcije treba ostati na stogu, a sve ostalo tijekom izračunavanja (uključujući sve operatore i operande) treba treba biti uklonjeno.

Za pretvorbu cijelog broja (int) u niz znakova koristiti funkciju čiji je prototip:

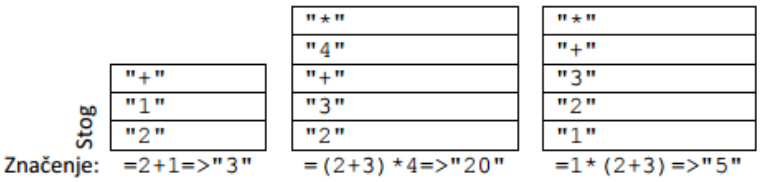
```
char *intustr(int val);
```

Za pretvorbu niza znakova u cijeli broj (int) koristiti funkciju čiji je prototip:

```
int struint(char *str);
```

Funkcije za pretvorbu nije potrebno pisati.

Primjeri stogova sa značenjem izraza:



Napomene: Rješenja koja neće koristiti zadane funkcije za rad sa stogom donijet će manje bodova.

Zadatak 4. (14 bodova)

```
void izracunaj(Stog *stog){
        char e[10], e1[10]={0}, e2[10]={0};
        int rez;
        if(skini(e, stog)){
                izracunaj(stog);
                if(isdigit(e[0])) dodaj(e,stog);
                else{
                        skini(e1, stog);
                        skini(e2, stog);
                        if(e[0]=='+') rez = struint(e1)+struint(e2);
                        else rez = struint(e1)*struint(e2);
                        dodaj(intustr(rez), stog);
                }
        }
}
```

Zadatak 5. (14 bodova)

Prikažite sadržaj stoga u trenutku neposredno prije prvog izvođenja naredbe **return 0** u funkciji **f2** i naznačite broj bajtova koje zauzima svaka od varijabli.

```
int f2(double *polje2, int n2, int d) {
	int i, brojac=0;
	if (n2<=0){
		return 0;
	}
	for(i=1;i<=n2/d;i++){
		brojac+=f2(polje2+i+d,n2-i-d, d);
	}
	return brojac+1;
}
void f1(double *polje1, int n1){
	f2(polje1,n1, 3);
}
int main() {
	double polje[8];
	...
	f1(polje, 8);
}
```

Zadatak 5. (14bodova)

brojac	4
i	4
Povr. f2	4
polje2	4
n2=0	4
d=3	4
brojac	4
i	4
Povr. f2	4
polje2	4
n2=4	4
d=3	4
brojac	4
i	4
Povr. f2	4
polje2	4
n2=8	4
d=3	4
Povr. f1	4
polje1	4
n1=8	4

Zadatak 1. (13 bodova)

Za tip podatka **Stog** koji je ostvaren jednostruko povezanom listom definirane su funkcije za inicijalizaciju stoga, dodavanje jednog elementa na stog te uklanjanje jednog elementa sa stoga. Prototipovi navedenih funkcija su:

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

Funkcije **dodaj** i **skini** vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače. Napišite funkciju čiji je prototip:

```
int preurediStog(Stog *stog);
```

koja će promijeniti poredak elemenata u zadanom početnom stogu tako da najveći element stavi na vrh stoga, dok se poredak ostalih elemenata ne mijenja. Ako ima više jednakih, a najvećih, elemenata na vrh ih treba staviti sve, ali njihov poredak nije važan. Funkcija mora vratiti broj najvećih elementa.

Primjer: Stog s elementima (od vrha prema dnu) **3.1, 2.7, 4.5, 4.8, 4.5, 3.1, 4.8, 4.1** funkcija će promijeniti u **4.8, 4.8, 3.1, 2.7, 4.5, 4.5, 3.1, 4.1.**

Zad 1.

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

int preurediStog(Stog *stog){
    Stog pom1, pom2;
    double element, max;
    int brPojavljivanja = 0;
    /*Inicijaliziraj stogove*/
    init_stog(&pom1);
    init_stog(&pom2);
    /*Trazimo najveci element - prvi element na stogu proglasavamo najvecim*/
    if(skini(&element, stog)){
        max = element;
        brPojavljivanja = 1; /*Broj povljivanja je 1*/
        dodaj(element, &pom1);
    }else{
        /*Stog je prazan*/
        return 0;
    }
    /*Provjera ostalih elemenata na stogu*/
    while(skini(&element, stog)){
        if(element > max){
            max = element;
            brPojavljivanja = 1; /*Novi najveci element - broj pojavljivanja je 1*/
        }else if(element == max){
            brPojavljivanja++;
        }
        dodaj(element, &pom1);
    }
    /*Prvo u stog zapisujemo elemente koji su manji od najveceg*/
    while(skini(&element, &pom1)){
        if(element == max){
            /*Sva pojavljivanja najveceg elementa spremamo na pom2 stog*/
            dodaj(element, &pom2);
            continue;
        }
        dodaj(element, stog);
    }
    /*Sada na vrh stoga dodajemo sva pojavljivanja najveceg elementa*/
    while(skini(&element, &pom2)){
        dodaj(element, stog);
    }
    return brPojavljivanja;
}
```

Zadatak 4. (8 bodova)

Za tip podatka `Stog` koji je realiziran jednostruko povezanom listom definirane su funkcije za inicijalizaciju stoga, dodavanje elementa na stog te skidanje elementa sa stoga. Zadana su dva stoga `stog1` i `stog2`, a elementi stogova su podaci tipa `Osoba` koji sadrže informaciju o šifri osobe (cijeli broj) i visini (cijeli broj). Stog `stog1` sadrži podatke samo za muškarce, a `stog2` samo za žene i na oba stoga su podaci sortirani po visini tako da je najviša osoba na vrhu stoga. Prototipovi navedenih funkcija su:

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

Funkcije **`dodaj`** i **`skini`** vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače.

a)*(1 bod)* Napišite **definicije** odgovarajućih struktura (tipovi podataka **`Osoba`**, **`Stog`** i **`atom`**) za stog realiziran jednostruko povezanom listom.

b)*(7 bodova)* Napišite **rekurzivnu** funkciju **`spoji`** koja će sa stoga `stog2` skinuti sve podatke i staviti ih na stog `stog1`, ali tako da ostane očuvan poredak po visini. Poredak osoba s istom visinom nije bitan.

Primjer:

	stog1		**stog2**		**stog1 (nakon spoji)**
vrh stoga->	345, 202	vrh stoga->	348, 203	vrh stoga->	348, 203
	161, 200		234, 200		345, 202
	654, 197		111, 189		234, 200
	122, 182		981, 169		161, 200
	234, 170				654, 197
					111, 189
					122, 182
					234, 170
					981, 169

a)(1bod)

```
typedef struct {

      int sifra;
      int visina;
} Osoba;

struct at {
  Osoba element;
  struct at *sljed;
};
typedef struct at atom;
typedef struct{
      atom *vrh;
} Stog;
```

b) (7 bodova)

```
void spoji (Stog *s1, Stog *s2){
  Osoba os1, os2, pom;
  int imas1, imas2;
  imas1=skini(&os1, s1);
  imas2=skini(&os2, s2);
  if (!imas1 && !imas2) return;
  else if (imas1 && imas2){
    if (os1.visina <os2.visina)  {
      dodaj (os1,s1);
      pom=os2;
    }
    else{
      dodaj (os2,s2);
      pom=os1;
    }
  }
  else if (imas1)
    pom=os1;
  else
    pom=os2;

  spoji (s1,s2);
  dodaj (pom,s1);
}
```

4. zadatak (11 bodova)

Zadan je tip podatka Stog za koji su definirane funkcije za inicijalizaciju stoga, dodavanje elementa na stog te za brisanje elementa sa stoga. Prototipovi navedenih funkcija su

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

Funkcije dodaj i skini vraćaju 1 ako je operacija dodavanja, odnosno skidanja uspjela, a 0 inače.
Napišite funkciju **bezEkstrema** koja će sa zadanog stoga izbaciti sva pojavljivanja najmanjeg i najvećeg elementa. Međusobni poredak ostalih elemenata ne smije se mijenjati. Rješenje treba biti neovisno o implementaciji apstraktnog tipa podataka Stog. Smiju se koristiti samo zadane funkcije i eventualno pomoćni stogovi.

4. (11 bodova)

```
void bezEkstrema(Stog *stog) {
	int elemMin, elemMax, stavka;
	Stog pomStog;

	if(!skini(&stavka, stog)) return; // nema se što raditi, stog je prazan

	// proglasi prvi element stoga trenutno najvećim i najmanjim
	elemMin=stavka; elemMax=stavka;

	init_stog(&pomStog);   // iniciraj pomoćni stog i stavi prvi element na njega
	dodaj(stavka,&pomStog);

	while(skini(&stavka,stog)){ // dok je elemenata
		if (stavka>elemMax) elemMax=stavka;  // provjeri je li novi možda najveći
		if (stavka<elemMin) elemMin=stavka;  // ili najmanji
		dodaj(stavka,&pomStog); // i stavi ga na pomoćni stog
	}

	while(skini(&stavka,&pomStog)){ // s pomoćnog stoga prepiši na originalni
		// sve osim ekstrema
		if ((stavka<elemMax) && (stavka>elemMin)) dodaj(stavka,stog);
	}
}
```