Zadaci za vježbu – Liste i redovi

1. (70 bodova) U red realiziran jednostruko povezanom listom spremaju se podaci cjelobrojnog tipa (**int**). Napisati funkcije za stavljanje u red i skidanje iz reda. Funkcije trebaju vratiti 1 ako je operacija uspješno obavljena, a 0 inače.

Napisati dodatnu funkciju koja će sve elemente zadanog reda slučajno rasporediti u dva nova reda. Svi redovi trebaju biti realizirani jednostruko povezanom listom. Prilikom realizacije dodatne funkcije nije dopušteno izravno pristupati elementima redova, već treba koristiti ranije napisane funkcije za stavljanje u red i skidanje iz reda. Nove, slučajno napunjene redove, treba vratiti u glavni program.

```
typedef struct z{
    int element;
    struct z *sljed;
}zapis;

int stavi(zapis **ulaz, zapis **izlaz, int e) {
    zapis *novi;
    novi = (zapis *)malloc(sizeof(zapis));
    if (novi == NULL) return 0;

    novi->element = e;

    novi->sljed = NULL;
    if (*izlaz == NULL) *izlaz = novi;
    else (*ulaz)->sljed = novi;

    *ulaz = novi;
    return 1;
}

int skini(zapis **ulaz, zapis **izlaz, int *e) {
    zapis *pom;

    if (*izlaz == NULL) return 0;

    pom = *izlaz;
    *e = pom->element;
    *izlaz = pom->sljed;
    if (*izlaz == NULL) *ulaz = NULL;
    free(pom);

    return 1;
}

void fun3(zapis **u1, zapis **i1, zapis **u2, zapis **i2, zapis **u3, zapis
**i3) {
    int e;

    srand((unsigned int)time(NULL));

    while (skini(u1, i1, &e)) {
        if (rand()%2) stavi(u2, i2, e);
        else stavi(u3, i3, e);
    }
}
```

2. U red realiziran **dvostruko** povezanom listom spremaju se cjelobrojni podaci (**long**). Napisati funkciju za skidanje jednog podatka iz reda i funkciju za stavljanje novog podatka u red. Napisati glavni program u kojem će se, koristeći prije napisanu funkciju, u red staviti brojevi od 1 do 10.

```
typedef struct z {
    long el;
    struct z *sljed;
    struct z *preth;
} zapis;

int stavi(zapis **ulaz, zapis **izlaz, int elm) {

    zapis *novi = (zapis *)malloc(sizeof(zapis));
    if (!novi) return 0;

    novi->el = elm;
    novi->sljed = NULL;

    if (*ulaz == NULL) {
        *ulaz = *izlaz = novi;
        novi->preth = NULL;
    }
    else {
        (*ulaz)->sljed = novi;
        novi->preth = *ulaz;
        *ulaz = novi;
    }

    return 1;
}

int skini(zapis **ulaz, zapis **izlaz, int *elm) {

    zapis * pom;
    if (*ulaz == NULL) return 0;  // može i *izlaz == NULL

    pom = *izlaz;
    *izlaz = (*izlaz)->sljed;
    if (*izlaz == NULL) *ulaz = NULL;
    else (*izlaz)->preth = NULL;

    *elm = pom->el;
    free(pom);
    return 1;

}

void main() {
    zapis *ulaz, *izlaz;
    long i;

    ulaz = izlaz = NULL;
    for (i=1; i<=10; i++) {
        stavi(&ulaz, &izlaz, i);
    }
}
```

3. U jednostruko povezanu listu spremaju se podaci o jelima na meniju u nekom restoranu: šifra jela (**int**), naziv jela (**50+1 znak**), cijena (**float**) i opis jela (**300+1 znak**). Lista je sortirana uzlazno, prema cijeni jela. Napisati funkciju koja će u listu dodati podatke o novom jelu (lista mora ostati sortirana). Funkcija treba imati prototip:

```
int dodaj(zapis **glava, int sif, char naz[], float cijena, char opis[]);
```

```
typedef struct el {
    int sifra;
    char naziv[50+1];
    float cijena;
    char opis[200+1];
} element;

typedef struct z {
    element el;
    struct z *sljed;
} zapis;

int dodaj(zapis **glava, int sif, char naz[], float cijena, char opis[]) {
    zapis *novi;

    novi = (zapis *)malloc(sizeof(zapis));
    if (novi == NULL) return 0;
    novi->el.sifra = sif;
    strcpy(novi->el.naziv, naz);
    novi->el.cijena = cijena;
    strcpy(novi->el.opis, opis);

    while (*glava && (*glava)->el.cijena < cijena) {
        glava = &((*glava)->sljed);
    }

    novi->sljed = *glava;
    *glava = novi;

    return 1;
}
```

4. U dvostruko povezanu listu spremaju se cjelobrojni podaci (**long**). Napisati funkciju koja će imati prototip:

```
int izbaciN(zapis **glava, zapis **rep, int N)
```

i koja će iz liste izbaciti prvih **N** elemenata. Ako u listi ima manje od **N** elemenata, funkcija ih treba izbaciti sve. Funkcija treba vratiti stvarni broj izbačenih elemenata.

```
int izbaciN(zapis **glava, zapis **rep, int N) {
    int i = 0;
    zapis *pom;
    while (*glava && i<N) {
        pom = *glava;
        *glava = pom->sljed;
        free(pom);
        i++;
    }

    if (*glava == NULL) *rep == NULL;
    else (*glava)->preth = NULL;

    return i;
}
```