Strukture podataka i algoritmi 1

Popravni test II (maksimalno 13 poena)

Ime i prezime	Indeks	Broj poena

1. Definisati strukturu **Knjiga** koja sadrži **naslov**, **autora**, **godinu izdanja** i **cenu**. Napisati funkciju koja za dati niz knjiga i njegovu dužinu vraća **pokazivač** na najstariju knjigu.

```
// jedno moguće rešenje
typedef struct Knjiga {
    char naslov[100];
    char autor[100];
    int godina;
    double cena;
} Knjiga;


Knjiga *najstarija(Knjiga *niz, int n) {
    if (n <= 0) return NULL;
    Knjiga *min = &niz[0];
    for (int i = 1; i < n; i++)
        if (niz[i].godina < min->godina)
            min = &niz[i];
    return min;
}
```

2. Dati odgovore na sledeća pitanja:

a) Za deklaraciju `int m[3][4]`, napisati izraz koji koristi **isključivo pokazivačku aritmetiku** (***** i **+**), a koji je ekvivalentan izrazu `m[i][j]`.

```
*(*(m + i) + j)
```

b) Šta je rezultat sledećeg koda:

```
int v[5] = {4, 8, 15, 16, 23};
int *p = v + 1;
printf("%d %d", *(p + 2), *p + 2);
```

16 10

c) Šta je rezultat sledećeg koda:

```
char s[] = "PROGRAM";
char *p = s + 2;
printf("%c %c %s", *p, *(p + 3), p);
```

O A OGRAM

d) Šta je rezultat sledećeg koda:

```
int a[2][3] = {{10, 20, 30}, {40, 50, 60}};
int *p = a[1];
printf("%d", *(p + 0) + *(a[0] + 2));
```

70

3. Neka je dat sledeći niz celih brojeva:

7	3	9	2	5	8

a) Demonstrirati rad **insertion sort** algoritma sortiranjem niza neopadajuće. Prikazati stanje niza posle svakog koraka algoritma.

7 3 9 2 5 8

3 7 9 2 5 8

3 7 9 2 5 8

2 3 7 9 5 8

2 3 5 7 9 8

2 3 5 7 8 9

b) Koliko poređenja izvrši insertion sort ako je ulazni niz već sortiran neopadajuće i zašto?
n - 1 poređenja: svaki element se poredi samo sa svojim levim susedom, utvrđuje se da je već na mestu i ne pomera se. To je najbolji slučaj, složenosti O(n)

c) Porediti algoritme **bubble sort**, **merge sort** i **insertion sort** po vremenskoj kompleksnosti u najgorem slučaju, od najboljeg ka najgorem i obrazložiti.

Merge sort je najbolji jer mu je i u najgorem slučaju kompleksnost O(n log n) (zbog rekurzivnog deljenja niza na pola i linearnog spajanja), dok su bubble sort i insertion sort lošiji i međusobno jednaki sa $O(n^2)$, jer oba u najgorem slučaju za svaki element prolaze kroz gotovo ceo niz (ugnežđene petlje).

4. Napisati rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int f(int x) { return x + 1; }
int g(int x) { return x * 2; }
int h(int x) { return x - 3; }

int (*izaberi(int k))(int) {
    int (*tabela[3])(int) = { f, g, h };
    return tabela[k % 3];
}

int main(void) {
    int x = 4;
    for (int i = 0; i < 4; i++)
        x = izaberi(i)(x);
    printf("%d\n", x);
    return 0;
}
```

8

5. Napisati funkciju koja obrće (reverse) jednostruko povezanu listu celih brojeva tako što menja veze između postojećih čvorova (bez alokacije novih čvorova; potrebno je prevezati pokazivače liste) i vraća pokazivač na novu glavu liste. Koristi se struktura iz 6. zadatka.

```
struct cvor *obrni(struct cvor *glava) {
    struct cvor *prethodni = NULL;
    while (glava != NULL) {
        struct cvor *sledeci = glava->sledeci;
        glava->sledeci = prethodni;
        prethodni = glava;
        glava = sledeci;
    }
    return prethodni;
}
```

6. Za datu jednostruko povezanu listu celih brojeva napisati funkciju koja vraća vrednost čvora koji se nalazi tačno na sredini liste. Ukoliko lista ima paran broj čvorova, vratiti vrednost drugog od dva srednja čvora. (Primeri: 1→2→3→4→5 daje 3; 1→2→3→4 daje 3.)

```
struct cvor {
   int vrednost;
   struct cvor *sledeci;
}
```

```
int sredina(struct cvor *glava) {
    struct cvor *spor = glava, *brz = glava;
    while (brz != NULL && brz->sledeci != NULL) {
        spor = spor->sledeci;
        brz = brz->sledeci->sledeci;
    }
    return spor->vrednost;
}
```

7. Date su dve jednostruko povezane liste celih brojeva. Napisati funkciju koja kreira novu listu naizmeničnim uzimanjem elemenata: prvi element prve liste, prvi element druge, drugi element prve, drugi element druge, itd. Kada se jedna lista isprazni, na kraj nadovezati sve preostale elemente druge liste. Koristi se struktura iz 6. zadatka. Novu listu kreirati prevezivanjem pokazivača na elemente postojećih lista.

Primer: A = 1->2->3, B = 7->8 -> rezultat je 1->7->2->8->3

```
struct cvor *prevezi(struct cvor *l1, struct cvor *l2) {
    if (l1 == NULL) return l2;
    if (l2 == NULL) return l1;

    struct cvor *nova = NULL, *novaPom = NULL;
    int redosled = 1;

    while (l1 != NULL && l2 != NULL) {
        struct cvor *tekuci;
        if (redosled == 1) {
            tekuci = l1;
            l1 = l1->sledeci;      // pomeri izvor PRE prevezivanja
            redosled = 2;
        } else {
            tekuci = l2;
            l2 = l2->sledeci;
            redosled = 1;
        }

        if (nova == NULL) nova = tekuci;
        else novaPom->sledeci = tekuci;
        novaPom = tekuci;          // rep je uvek upravo dodati cvor
    }

    novaPom->sledeci = (l1 != NULL) ? l1 : l2;
    return nova;
}
```

8. Napisati program kod koga se preko komandne linije zadaje jedan ceo broj n. Potrebno je dinamički alocirati „stepenasti“ niz (niz pokazivača) u kome red i (za i = 0, 1, ..., n-1) ima tačno i + 1 elemenata. Niz popuniti vrednostima Paskalovog trougla (svaki rubni element je 1, a svaki unutrašnji je zbir dva elementa iznad njega) i ispisati ga. Na kraju osloboditi svu zauzetu memoriju.

Primer: n = 3

1		
1	1	
1	2	1

Primer: n = 5

1				
1	1			
1	2	1		
1	3	3	1	
1	4	6	4	1

Primer: n = 7

1						
1	1					
1	2	1				
1	3	3	1			
1	4	6	4	1		
1	5	10	10	5	1	
1	6	15	20	15	6	1

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

int main(int argc, char *argv[]) {
    if (argc < 2) { printf("Nije zadat broj n.\n"); return 1; }
    int n = atoi(argv[1]);
    if (n <= 0) { printf("n mora biti pozitivan.\n"); return 1; }

    int **t = malloc(n * sizeof(int *));
    for (int i = 0; i < n; i++)
        t[i] = malloc((i + 1) * sizeof(int));   // red i ima i+1 elemenata

    for (int i = 0; i < n; i++) {
        t[i][0] = t[i][i] = 1;
        for (int j = 1; j < i; j++)
            t[i][j] = t[i-1][j-1] + t[i-1][j];
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++)
            printf("%d ", t[i][j]);
        printf("\n");
    }
    for (int i = 0; i < n; i++) free(t[i]);
    free(t);
    return 0;
}
```