Strukture podataka i algoritmi 1

Test II (maksimalno 13 poena)

Ime i prezime	Indeks	Broj poena

1. Dopuniti delove koda tako da celokupan kod ispravno učitava i ispisuje podatke o strukturi životinja.

```
#include <stdio.h>

typedef struct Animal {
    char type[100];
    char name[100];
    double weight;
    int age;
} Animal;

void getAnimalDetails(Animal *animal) {
    scanf("%s %s %lf %d", animal->type, animal->name, &animal->weight, &animal->age);
}

int main(void) {
    Animal animal;
    getAnimalDetails(&animal);

    printf("%s %s %lf %d\n", animal.type, animal.name, animal.weight, animal.age);

    return 0;
}
```

2. Dati odgovore na sledeća pitanja:

a) Šta je rezultat sledećeg koda:
```
int a[2][2] = {3, 6, 9, 12};
printf("%d", *(a[0]+2));
```
9

b) Zaokružiti izraze/izraz kojima treba zameniti x tako da izraz **a[i][j][k][l]**=x bude tačan:
   a. ((((a+i)+j)+k)+l)
   b. *(*(*(*(a+i)+j)+k)+l)
   c. (((a+i)+j)+k+l)
   d. ((a+i)+j+k+l)

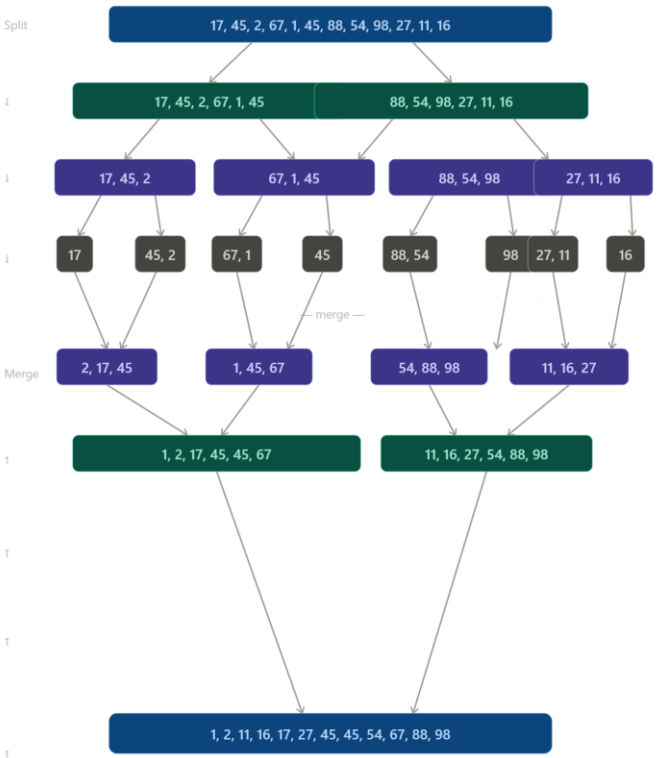
c) Neka je deklarisana promenljiva na sledeći način: int **a[3][4]**. Šta predstavlja izraz ***(a+2)**?
Pokazivač na prvi element trećeg reda matrice.

d) Šta je rezultat sledećeg koda:
```
int a[4][5] = {
      {12,  5,  8, 17,  3},
      {21, 14,  9,  6, 11},
      { 7, 18, 25, 13,  4},
      {10, 16, 22,  1, 19}
   };
printf("%d", *(*(*(a + 2) + 3)));
```
Kompajlerska greška

3. Neka je dat sledeći niz celih brojeva: 17 , 45 , 2 , 67 , 1 , 45 , 88 , 54 , 98 , 27 , 11 , 16

Na primeru ovog niza demonstrirati rad **merge sort** algoritma za sortiranje gde je niz potrebno sortirati neopadajuće. Prikazati svaki korak rada algoritma. Dati komentar o poređenju vremenske kompleksnosti ovog algoritma i algoritma **bubble sort**.

Vizuelno rešenje:



Merge sort je značajno efikasniji od bubble sorta — dok bubble sort ima vremensku kompleksnost O(n²) u prosečnom i najgorem slučaju, merge sort garantuje O(n log n) u svim slučajevima, što ga čini daleko boljim izborom za veće skupove podataka, uz jedini kompromis da zahteva dodatnih O(n) memorije.

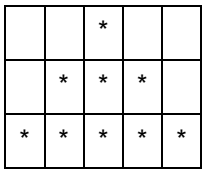
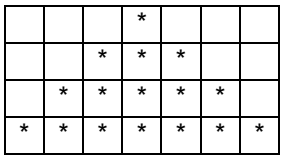
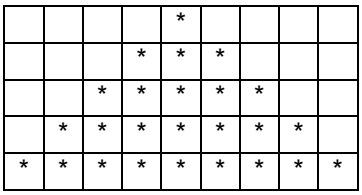
4. Napisati rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int add(int a, int b) { return a + b; }
int mul(int a, int b) { return a * b; }
int sub(int a, int b) { return a - b; }
int apply(int (*op)(int, int), int x, int y) { return op(x, y); }

int func(int (*ops[3])(int,int), int x, int y) {
    int r = x;
    for (int i = 0; i < y; r = apply(ops[i], r, y), i++);
    return r;
}

int main(void) {
    int (*ops[3])(int,int) = { mul, add, sub };
    printf("%d\n", func(ops, 2, 3));
    return 0;
}
```

6

5. Napisati program kod koga se preko komandne linije zadaje jedan ceo broj n (neparan broj). U zavisnosti od zadatog broja, potrebno je kreirati matricu karaktera koja je popunjena zvezdicama u

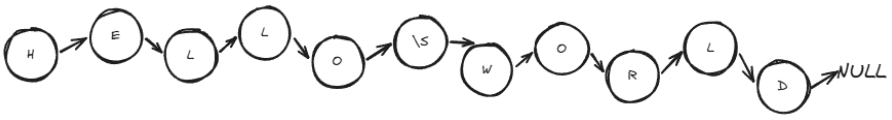
obliku trougla. Prostor za matricu karaktera **obavezno** alocirati na adekvatan način i na kraju **osloboditi**. Prazna mesta u matrici se mogu popunjavati space karakterima.

Primer: n = 5	Primer: n = 7	Primer: n = 9
		

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

int main(int argc, char *argv[]) {

    if (argc != 2) {
        fprintf(stderr, "Upotreba: %s <neparan_broj>\n", argv[0]);
        return EXIT_FAILURE;
    }
    int n = atoi(argv[1]);

    if (n <= 0 || n % 2 == 0) {
        fprintf(stderr, "Greška: n mora biti pozitivan neparan broj.\n");
        return EXIT_FAILURE;
    }

    int rows = (n + 1) / 2;
    int cols = n;
    // Alokacija pokazivaca na vrste
    char **matrica = (char **)malloc(rows * sizeof(char *));
    if (!matrica) {
        fprintf(stderr, "Greška: alokacija memorije nije uspela.\n");
        exit(EXIT_FAILURE);
    }
    // Alokacija svake vrste
    for (int i = 0; i < rows; i++) {
        matrica[i] = (char *)malloc(cols * sizeof(char));
        if (!matrica[i]) {
            fprintf(stderr, "Greška: alokacija memorije nije uspela.\n");
            exit(EXIT_FAILURE);
        }
        memset(matrica[i], ' ', cols);
    }
    // Popunjavanje
    for (int i = 0; i < rows; i++) {
        int zvezde  = 2 * i + 1;
        int pocetak = (n - zvezde) / 2;

        for (int j = pocetak; j < pocetak + zvezde; j++) {
            matrica[i][j] = '*';
        }
    }

    // Brisanje
    for (int i = 0; i < rows; i++) {
        free(matrica[i]);
    }
    free(matrica);
}
```

6. Za dati pokazivać na jednostruko povezanu listu karaktera (ne dužu od 100 čvorova), napisati funkciju koja vraća string koji se formira dodavanjem karaktera iz svakog čvora liste. Struktura koja se koristi za listu je sledeća:

```
struct cvor {
    char karakter;
    struct cvor *sledeci;
};
```



Za dati primer, rezultat poziva funkcije je string „HELLO WORLD“.

```
char *lista_u_string(cvor *glava)
{
    char *s = malloc(101 * sizeof(char));
    if (s == NULL){
        printf("Greska prilikom alokacije memorije\n");
        exit(1);
    }
    int i = 0;
    while (glava != NULL && i < 100)
    {
        s[i++] = glava->karakter;
        glava = glava->sledeci;
    }
    s[i] = '\0';
    return s;
}
```

7. Neka su date dve jednostruko povezane liste celih brojeva koje su, svaka za sebe, sortirane neopadajući. Napisati funkciju koja prihvata pokazivače na obe liste i kreira novu listu tako što spaja ove dve liste u jednu, takođe neopadajuće sortiranu listu.

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

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

    struct cvor *head = (l1->vrednost <= l2->vrednost) ? l1 : l2;
    struct cvor *tail = head;

    if (l1 == head) l1 = l1->sledeci;
    else            l2 = l2->sledeci;

    while (l1 && l2)
    {
        if (l1->vrednost <= l2->vrednost)
            { tail->sledeci = l1; l1 = l1->sledeci; }
        else
            { tail->sledeci = l2; l2 = l2->sledeci; }
        tail = tail->sledeci;
    }

    tail->sledeci = l1 ? l1 : l2;
    return head;
}
```

8. Napisati funkciju koja za datu jednostruko povezanu listu karaktera utvrđuje da li je lista palindrom ili nije. Lista je, slično rečima, palindom ukoliko se „čita“ isto sa leve i desne strane. Struktura koja se koristi za predstavljanje liste je ista kao i u 6. zadatku. Možete podrazumevati da u listi neće biti drugih karaktera osim malih slova i cifara.

```
int palindrom(struct cvor *glava) {
    char *string = lista_u_string (glava); // lista_u_string je funkcija iz 6. zadatka
    int duzina = strlen(string);
    int i = 0, j = duzina-1;
    while (i < j) {
        if (string[i++] != string[j--]) return 0;
    }
    return 1;
}
```