REKURZIJA - zadaci

1. **Napisati rekurzivnu funkciju koja računa najveću cifru datog celog broja i program koji testira rad funkcije.**

```
#include <stdio.h>
int veci(int a, int b)
{
 if (a > b) return a;
 else return b;
}

int najveca(int n)
{
 if (n<10) return n;
 else return veci(najveca(n/10), n%10);
}

int main()
{
 int n;

 printf("Unesi broj\n");
 scanf("%d", &n);

 printf("Najveca cifara je: %d\n", najveca(n));

 return 0;
}
```

2. **Napisati rekurzivnu funkciju koja računa n-ti element u Fibonačijevom nizu. Popraviti funkciju tako da se problemi manje dimenzije rešavaju samo jedanput. (MEMOIZACIJA).**

Analiza rešenja
U slučaju da se rekurzijom problem svodi na više manjih podproblema koji se mogu preklapati, postoji opasnost da se pojedini podproblemi manjih dimenzija rešavaju veći broj puta. Na primer,
fibonacci(20) = fibonacci(19) + fibonacci(18)
fibonacci(19) = fibonacci(18) + fibonacci(17)
tj. problem fibonacci(18) se resava dva puta. Problemi manjih dimenzija će se rešavati još veći broj puta.
Rešenje za ovaj problem je kombinacija rekurzije sa tzv. "dinamičkim programiranjem" tj. podproblemi se rešavaju samo jednom, a njihova rešenja se pamte u memoriji (obično u nizovima ili matricama), odakle se koriste ako tokom rešavanja ponovo budu potrebni.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 50
/* Niz koji cuva resenja podproblema. */
```

```
int f[MAX];
/* Funkcija izracunava n-ti fibonacijev broj */
int fibonacci(int n)
{
/* Ako je podproblem vec resen, uzimamo gotovo resenje! */
if(f[n] != 0)
return f[n];
/* Izlaz iz rekurzije */
if(n < 2)
return f[n] = 1;
else
/* Rekurzivni pozivi */
return f[n] = fibonacci(n - 1) + fibonacci(n - 2);
}
/* Test program */
int main()
{
int n, i;
/*Inicijalizuje se niz*/
for(i=0; i<MAX; i++)
f[i] = 0;
scanf("%d", &n);
printf("%d\n", fibonacci(n));
return 0;
}
```

3. **Napisati**
a) rekurzivnu b) iterativni verziju
funkcija koje racunaju n-ti clan Fibonacijevog niza
Napisati i c) rekurzivnu verziju sa memoizacijom

```
#include <cstdio>
#define MAXN 1000010

long long fib(int n) { //O(1.6^n)   REKURZIVNO i NEEFIKASNO RESENJE
    if(n == 1 || n == 2) return 1;
    return fib(n-1) + fib(n-2);
}

int fibo[MAXN], f[MAXN];

long long it_fib(int n) {    //ITERATIVNO RESENJE, EFIKASNIJE OD PRETHODNOG
    fibo[1] = fibo[2] = 1;
    for(int i = 3; i <= n; i++) fibo[i] = fibo[i-1] + fibo[i-2];
    return fibo[n];
}


long long fibonacci(int n)  //REKURZIVNO EFIKASNIJE RESENJE SA MEMOIZACIJOM
{
/* Ako je podproblem vec resen, uzimamo gotovo resenje! */
if(f[n] != 0)
```

```
return f[n];
/* Izlaz iz rekurzije */
if(n < 2)
return f[n] = 1;
else
/* Rekurzivni pozivi */
return f[n] = fibonacci(n - 1) + fibonacci(n - 2);
}
 using namespace std;

int main()
{
   int n; scanf("%d", &n);
   //printf("%lld", it_7fib(n));
  //   printf("%lld", fibonacci(n));
       printf("%lld", fib(n));
   return 0;
}
```

4. Napisati rekurzivnu funkciju koja izračunava vrednost binomnog koeficienta $\binom{n}{k}$

```
#include <stdio.h>
int binom(int n, int k)
{
 if (k == 0) return 1;
 else if (n == 0) return 0;
 else return binom(n-1, k-1) + binom(n-1, k);
}

int main()
{
 int n, k;

 printf("Unesi broj n\n");
 scanf("%d", &n);

 printf("Unesi broj k\n");
 scanf("%d", &k);

 printf("Vrednost binomnog koeficijenta (n nad k) je: %d\n", binom(n, k));

 return 0;
}
```

5.

(a) Napisati rekurzivnu funkciju koja prikazuje dekadne cifre datog celog broja.

(b) Napisati rekurzivnu funkciju koja prikazuje dekadne cifre datog celog broja u obrnutom poretku.

```
#include <stdio.h>
void ispisi_cifre(int x)
{
```

```
 if (x < 10) printf("%d ", x);
 else
 {
 ispisi_cifre(x/10);
 printf("%d ", x%10);
 }
}
void ispisi_cifre2(int x)
{
 if (x < 10) printf("%d ", x);
 else
 {
 printf("%d ", x%10);
 ispisi_cifre2(x/10);
 }
}
int main()
{
 int n;

 printf("Unesi broj\n");
 scanf("%d", &n);

 printf("Cifre su: ");
 ispisi_cifre(n);
// ispisi_cifre2(n);
 printf("\n");

 return 0;
}
```

6.
Napisati rekurzivnu funkciju za odredivanje NZD dva broja Euklidovim algoritmom.

```
unsigned NZD(unsigned x, unsigned y)
{ return x==0? y:NZD(y%x,x); }
```

7. **Napisati rekurzivnu funkciju koja odredjuje minimum niza celih brojeva. Napisati program koji testira ovu funkciju, za niz koji se ucitava sa standardnog ulaza. Niz nece biti duzi od 256 i njegovi elementi se unose sve do kraja ulaza.**

Primer 1
Ulaz: 1 5 9 8 4 Izlaz: 1

```
#include <stdio.h>
#include <stdlib.h>
#define Nmax 256

int minimum(int a, int b)
{
    if (a<b) return a;
    else return b;
}
```

```
int min_niza(int a[], int n)
{
    if (n==0) return a[0];
    else
        return minimum(min_niza(a, n-1), a[n]);
}

int main()
{
    int a[Nmax], i=0;

    printf("Unesi elemente niza\n");

    while(1)
    {
        scanf("%d", &a[i]);
        if (feof(stdin)) break;
        i++;
    }

    if (i>0)
    printf("Minimum niza je %d\n", min_niza(a, i));
    else printf("Nije unet nijedan element\n");


    return 0;
}
```

8. **Napisati rekurzivnu funkciju koja sabira kubove dekadnih cifara celog broja x. Napisati program koji testira ovu funkciju, za broj koji se ucitava sa standardnog ulaza.**

Primer
Ulaz:
Unesite x: -1005
Izlaz: 126

```
#include <stdio.h>

int suma_cifara(long int n)
{   int cifra=n%10;
    int cifrakub=cifra*cifra*cifra;
        if (n/10)
                return cifrakub + suma_cifara(n/10);
    else
        return cifrakub;
}

int main()
{
 int br;
 printf("Unesite broj ");
 scanf("%d",&br);
 if (br<0) br=-br;
 printf("Suma kubova cifara je %d",suma_cifara(br));
 return 0;
}
```

9. Napisati rekurzivnu funkciju koja racuna broj elemenata u nizu a, koji su strogo manji od zadatog broja k. Napisati program koji testira ovu funkciju, za k i niz koji se zadaju sa standardnog ulaza. Prvo se unosi k, a zatim elementi niza a, sve do kraja ulaza. Pretpostaviti

da niz nece imati vise od 256 elemenata.

Primer
Ulaz: Unesite ceo broj: 5 Unesite elemente niza: 0 3 9 1 -8 19 34 5
Izlaz: 4

```
#include <stdio.h>
#define Nmax 256

int brojManjih(int a[], int k, int n)
{
    /* Izlazak iz rekurzije: za niz duzine jedan broj pojava broja x
    u nizu je 1 ukoliko je jedini element a[0] bas manji od x ili 0 inace */
    if (n == 1)
        return (a[0] < k) ? 1 : 0;

    /* U promenljivu bp se smesta broj pojave broja clanova manjih od k u prvih
n-1
    elemenata niza a. Ukupan broj pojavljivanja clanova manjih od  k u celom
    nizu a je jednak bp uvecanom za jedan ukoliko je se na
    poziciji n-1 u nizu a nalazi broj manji od k */
    int bp = brojManjih(a, k, n - 1);
    return (a[n - 1] < k) ? (1 + bp) : bp;
}

int main()
{
    int a[Nmax], i=0,  k;

    printf("Unesite k\n");
    scanf("%d", &k);

    printf("Unesite elemente niza:");

    while (scanf("%d", &a[i]) != EOF)
    {
        i++;
        if (i == Nmax)
            break;
    }



    printf("Trazeni rezultat je %d\n", brojManjih(a, k, i));

    return 0;
}
```

10.
Koristeći uzajamnu (posrednu) rekurziju napisati:
(a) funkciju unsigned paran(unsigned n) koja proverava da li je broj cifara broja x paran i vraća 1 ako jeste, a 0 inače;
(b) i funkciju unsigned neparan(unsigned n) koja proverava da li je broj cifara broja x neparan i vraća 1 ako jeste, a 0 inače.
Napisati program koji testira napisane funkcije tako što za heksadekadni broj koji se unosi sa standardnog ulaza ispisuje da li je broj njegovih cifara paran ili neparan.

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

```c
/* NAPOMENA: Ovaj problem je iskoriscen da ilustruje uzajamnu
   (posrednu) rekurziju */

/* Deklaracija funkcije neparan mora da bude navedena jer se ta
   funkcija koristi u telu funkcije paran, tj. koristi se pre svoje
   definicije. Funkcija je mogla biti deklarisana i u telu funkcije
   paran. */
unsigned neparan(unsigned n);

/* Funkcija paran vraca 1 ako broj n ima paran broj cifara, inace
   vraca 0 */
unsigned paran(unsigned n)
{
  if (n <= 9)
    return 0;
  else
    return neparan(n / 10);
}

/* Funkcija neparan vraca 1 ako broj n ima neparan broj cifara,
   inace vraca 0 */
unsigned neparan(unsigned n)
{
  if (n <= 9)
    return 1;
  else
    return paran(n / 10);
}

int main()
{
  int n;

  /* Ucitava se ceo broj */
  scanf("%x", &n);

  /* Ispisuje se rezultat */
  printf("Uneti broj ima %sparan broj cifara.\n",
         (paran(n) == 1 ? "" : "ne"));

  return 0;
}
```