MI 2012.

Zadatak 2. *(5 bodova)*

a) Napišite rekurzivnu funkciju koja će tehnikom raspolavljanja izračunati i vratiti sumu članova polja.

b) Napišite glavni program u kojem ćete učitati elemente polja s tipkovnice. Broj elemenata polja nije unaprijed poznat, a učitavanje se prekida kada se učita broj 0. Za pohranu elemenata potrebno je koristiti dinamički alocirano polje, a elementi se smiju učitati samo jednom.

Napomena: Nerekurzivno rješenje neće se priznavati.

2. zadatak

```
int suma(int a[], int lijevo, int desno) {
        int sred = 0;
        if(lijevo > desno) {
                return 0;
        } else if(lijevo == desno){
                return a[lijevo];
        } else {
                sred = ((desno - lijevo) / 2) + lijevo;
                return a[sred] + suma(a,lijevo,sred-1) + suma(a,sred+1,desno);
        }
}
```

LJIR 2012.

2. zadatak (13 bodova)

Zadana je jednostruko povezana lista sljedećom strukturom:

```
typedef struct at{
        int element;
        struct at *sljed;
} atom;
```

Napisati rekurzivnu funkciju koja će promjenom pokazivača na sljedeće elemente obrnuti listu. Nerekurzivno rješenje neće se priznavati.

2. (13 bodova)

```
void obrniListu(atom** glava) {
        atom* prvi;
        atom* pom;

        if (*glava == NULL) return;

        prvi = *glava;
        pom = prvi->sljed;

        if (pom == NULL) return;

        obrniListu(&pom);

        prvi->sljed->sljed = prvi;
        prvi->sljed = NULL;

        *glava = pom;
}
```

MI 2013.

Zadatak 3. *(5 bodova)*

Neka je zadano polje **a** koje se sastoji od **n** cijelih brojeva. Napišite rekurzivnu funkciju **maxRekurzivno** prototipa:

```
int maxRekurzivno(int a[],int n, int *brojPojavljivanja)
```

koja će pronaći najveći element u polju **a** te njegovu učestalost (koliko puta se taj element pojavljuje u **a)**.

Napomena: Nerekurzivno rješenje neće se priznavati.

3.

```
int maxRekurzivno(int a[],int n, int *brojPojavljivanja)
{
	int kandidat;
	if (n==1){
		*brojPojavljivanja=1;
		return a[0];
	}
	kandidat=maxRekurzivno(a,n-1, brojPojavljivanja);
	if (kandidat>a[n-1]){
		return kandidat;
	}
	else if(kandidat==a[n-1]){
		(*brojPojavljivanja)++;
		return kandidat;
	}
	else{
		*brojPojavljivanja=1;
		return a[n-1];
	}
}
```

Najčešće
- L
u
n
- K
- N
n

DIR 2013.

Zadatak 2. (14 bodova)

Neka je zadano polje **a** koje se sastoji od **n** pozitivnih cijelih brojeva sortiranih silazno. Napišite rekurzivnu funkciju **postojiZbroj** koja će za zadani cijeli broj **m** vratiti 1 ako je **m** moguće napisati kao zbroj elemenata polja **a**, odnosno 0 ako to nije moguće. Podrazumijeva se da se elementi polja **a** mogu upotrijebiti samo po jednom.
Prototip funkcije **postojiZbroj** treba biti

`int postojiZbroj(int a[],int n, int m);`

Zad 2.

```
int postojiZbroj(int a[],int n, int m){
	int i, noviM;

	if(m<0) return 0;
	if (m==0) return 1;

	for(i=n-1; i>=0; i--){
		noviM = m - a[i];
		if (postojiZbroj(a,i,noviM)) return 1;
	}
	return 0;
}
```

MI 2014.

Zadatak 3. (5 bodova)

Svaki prirodni broj $N \geq 2$ može se napisati u obliku rastava na proste faktore:

$$N=p_1^{n1} \cdot p_2^{n2} \cdots p_k^{nk},$$

gdje su p_i, $i=1,\ldots,k$, različiti prosti brojevi (prosti faktori), a eksponenti $n_1,\ldots,n_k$ su prirodni brojevi.

Napišite rekurzivnu funkciju `najveci_eksponent` koja će za zadani broj `N` odrediti najveći eksponent s kojim se neki prosti broj pojavljuje u rastavu broja `N` na proste faktore (najveći od brojeva n_i). Prototip funkcije je:

`int najveci_eksponent (int n, int m);`

pri čemu je `m` trenutačni djelitelj (p_i). Početni poziv funkcije za zadani `n` je oblika `najveci_eksponent(n,2)`.

Primjeri:

Za `N` = 6, funkcija treba vratiti 1 jer $6 = 2^1 \cdot 3^1$.
Za `N` = 8, funkcija treba vratiti 3 jer $8 = 2^3$.
Za `N` = 36, funkcija treba vratiti 2 jer $36 = 2^2 \cdot 3^2$.

3. (5 bodova)

```
int najveci_eksponent(int n, int m){
	int dalje, eksponent=0;
	if (n<m) return 0; /* Osnovni slucaj */
	while(n % m == 0){   /* Koliko puta m dijeli n */
		eksponent++;
		n /= m;
	}
	dalje= najveci_eksponent (n,m+1); /* Rekurzivno napredovanje */

	if (dalje>eksponent) return dalje; /* Veci se vraca pozivajucoj proceduri */
	return eksponent;
}
```

MI 2015.

Zadatak 3. (5 bodova)

Napisati program koji pomoću rekurzivne funkcije nalazi, a u glavnom programu ispisuje najmanji zajednički višekratnik i najveći zajednički djelitelj dva prirodna broja. Brojevi se učitavaju putem standardnog ulaza u glavnom programu.

LJIR 2015.

Zadatak 3. (15 bodova)

Zadano je polje **a** s ukupno **n** elemenata čiji su elementi pozitivni cijeli brojevi. Potrebno je napisati rekurzivnu funkciju **minNepar** koja će naći najmanji neparni broj u polju **a**, ako takav postoji, a u suprotnom vraća -1. Prototip funkcije je

```
int minNepar(int a[], int n);
```

Nerekurzivno rješenje se neće priznavati.

Zadatak 3. (15 bodova)

```
int minNepar(int* a, int n) {
	int znamenka, vrati, min;
	if (n==0) return -1;		// ako nema elemenata polja, vrati -1

	if (a[n-1] % 2==1){		// ako je zadnji element polja neparan
		min=a[n-1];		// onda je trenutno najmanji
	}
	else min=-1;
	vrati=minNepar(a,n-1);		// što vraća funkcija?
	if (vrati==-1) return min; // ako nema neparnih, onda je trenutni najmanji, najmanji
	if (min==-1) return vrati; // ako je trenutno najmanji -1, onda nastavi s onim što je vraćeno

	if (min<vrati){			// ako obje vrijednosti postoje, usporedi ih i vrati što treba.
		return min;
	}
	else return vrati;
}
```

JIR 2015.

Zadatak 2. (15 bodova)

Napišite rekurzivnu funkciju `samoUzlazno` čiji je prototip:

```
int samoUzlazno(int n);
```

Funkcija za zadani pozitivni cijeli broj *n* vraća broj koji je načinjen od onih znamenaka broja *n* koje su (gledano slijeva na desno) u uzlaznom poretku, pri čemu se poredak znamenaka (kakav je bio u *n*) ne smije mijenjati.

Primjer: za broj 175739 treba vratiti 179, za broj 21003 treba vratiti 23, a za 2062 treba vratiti 26.

Zadatak 2. (15 bodova)

```
int samoUzlazno(int n){
	int tmp,zn;
	if(n<10) return n;

	zn=n % 10;
	tmp=samoUzlazno(n/10);
	if (tmp % 10 < zn) tmp=tmp*10+zn;
	return tmp;
}
```

DIR 2015.

Zadatak 2. (18 bodova)

Napravite rekurzivnu funkciju `samoNeparni` čiji je prototip

`long samoNeparni(long n);`

Funkcija za zadani pozitivni cijeli broj `n > 0` vraća broj `k` koji se dobije tako da se iz broja `n` izostave parne znamenke

Npr: **`samoNeparni(2062) = 0, samoNeparni (21000) = 1, samoNeparni (123456) = 135.`**

Zadatak 2. (18 bodova)

```
long samoNeparni(long n){
	long tmp;
	int zn;
	if (n<10){
		tmp=0;
		if (n % 2==1) tmp=n;
		return tmp;
	}
	zn=n % 10;
	tmp=samoNeparni(n/10);
	if(zn % 2==1) tmp=tmp*10+zn;
	return tmp;
}
```

MI 2016.

Zadatak 1. (6 bodova)

Napišite rekurzivnu funkciju prototipa:

int PascalovTrokut (int redak, int stupac);

koja određuje i vraća cijeli broj koji se nalazi u retku **redak** i stupcu **stupac** u Pascalovom trokutu. Prikaz Pascalovog trokuta u obliku matrice:

	0	**1**	**2**	**3**	**4**
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

itd.

6 = 3 + 3

Primjeri:

- za poziv PascalovTrokut(0, 0), funkcija treba vratiti 1
- za poziv PascalovTrokut(1, 1), funkcija treba vratiti 1
- za poziv PascalovTrokut(2, 1), funkcija treba vratiti 2
- za poziv PascalovTrokut(4, 2), funkcija treba vratiti 6 (primjer sa slike)

Napomena: nerekurzivno rješenje se neće priznati.

1. zadatak

```
int PascalovTrokut(int redak, int stupac) {
  int clan;
  if (stupac == 0 || redak == stupac) {
    clan = 1;
  }
  //else if (stupac == 1 || (stupac + 1) == redak) {
  //  clan = redak;
  //}
  else {
    clan = PascalovTrokut(redak - 1, stupac - 1) +
      PascalovTrokut(redak - 1, stupac);
  }
  return clan;
}
```

LJIR 2016.

Zadatak 3. (20 bodova)

Neka je zadan niz znakova X. Potrebno je preurediti elemente niza X i to tako da se napravi zamjena uzastopnih parova znakova. Napišite dvije rekurzivne funkcije, `zamijeniL` i `zamijeniD`, koje rade traženu promjenu, s time da `zamijeniL` radi promjene uzastopnih parova gledano s lijeva, a `zamijeniD` radi promjene uzastopnih parova gledano s desna. Prototipovi funkcija su:

```
void zamijeniD(char *A, int duljina); void zamijeniL(char *A, int duljina);
```

pri čemu vrijednost varijable duljina označava broj znakova u nizu X. Primjer: za `X="AbCdEfG"` primjena funkcije `zamijeniL` daje `"bAdCfEG"`, dok primjena funkcije `zamijeniD` daje `"ACbEdGf"`.

Napomena: nerekurzivno rješenje se neće priznati.

3. Zadatak

```
void zamijeniD(char *A, int duljina){
	char tmp;
	if (duljina == 1){
		return;
	}
	if (duljina == 2){
		tmp = A[0]; A[0] = A[1]; A[1] = tmp;
		return;
	}
	tmp = A[duljina - 1]; A[duljina - 1] = A[duljina - 2]; A[duljina - 2] = tmp;
    zamijeniD(A, duljina - 2);
}

void zamijeniL(char *A, int duljina){
	char tmp;
	if (duljina == 1){
		return;
	}
	tmp = A[0]; A[0] = A[1]; A[1] = tmp;
	if (duljina == 2){
		return; // bila su dva znaka i zamijenili smo ih
	}
	zamijeniL1(&A[2], duljina - 2); /*ove dvije linije mogu biti i obrnute*/
}
```

DIR 2016.

Zadatak 4. (18 bodova)

Napišite rekurzivnu funkciju **preuredi** koja za zadani pozitivni cijeli broj **n** vraća broj, koji je dobiven tako da se svako pojavljivanje neparnih znamenki (pojava jedne ili više uzastopnih neparnih znamenki) u broju **n** zamijeni točno jednom znamenkom 1.

Poredak parnih znamenaka (uključujući 0) se ne mijenja.

Prototip funkcije je:

```
long preuredi(long n);
```

Primjeri: `preuredi(52331)=121, preuredi(225307)=22101, preuredi(1333)=1`

Napomena: Nerekurzivna rješenja se neće priznavati.

4. zadatak (18 bodova)

```
long preuredi(long n){
	int m;
	long p;

	if (n < 10) { // ovo je kraj rekurzivnih poziva ...
		if (n % 2 == 0) return n;
		else return 1;
	}
	else p = preuredi(n / 10);

	m = n % 10; // znamenka jedinica ulaza...
	if (m % 2 == 0) { // ako je m paran, dodaj ga na kraj
		return p * 10 + m;
	}
	else { // ako je m neparan, pogledaj zadnju znamenku od p,
		if (p % 10 == 1) { // ako je to znamenka 1, ne radi ništa
			return p;
		}
		else { // ako nije, stavi 1 na kraj
			return p * 10 + 1;
		}
	}
}
```

MI 2017.

Zadatak 1. (7 bodova)

a) Napišite rekurzivnu funkciju koja za znakovni niz (*string*) proizvoljne duljine provjerava ima li svaka otvorena uglata zagrada u nizu (znak '[') pripadajuću zatvorenu uglatu zagradu (znak ']') i obratno. Funkcija treba vratiti 1, ako svaka otvorena zagrada u nizu ima pripadajuću zatvorenu zagradu, a 0 inače.
Pretpostavite da se znakovni niz sastoji samo od znakova '[' i ']'. Prototip funkcije je:

```
int provjeriZagrade(char *niz, int *zastavica);
```

Naputak: `zastavica` je pomoćna vrijednost koja se koristi za praćenje broja neuparenih zagrada.

Primjeri:
Za niz "**[**[[]]]" funkcija treba vratiti 0, jer zagrada **[** nema pripadajuću zatvorenu zagradu.
Za niz "]]**]**[[]]" funkcija treba vratiti 0, jer zagrada **]** nema pripadajuću otvorenu zagradu.
Za niz "[[[]]]]" funkcija treba vratiti 1, jer svaka otvorena zagrada ima pripadajuću zatvorenu zagradu.
Napomena: nerekurzivno rješenje se neće priznati.

b) Napišite odsječak glavnog programa u kojem se poziva funkcija iz a) dijela zadatka za niz "[[]".

Komentar uz zadatak: uparenost zagrada znači da svaka otvorena uglata zagrada (znak '[') ima pripadajuću zatvorenu uglatu zagradu (znak ']'), gdje nije bitno da je redoslijed zagrada matematički ispravan, tj. i niz " []" i niz "][" su ispravno zadani.

Rješenja:

1. zadatak (7 bodova)

```c
#include <stdio.h>
int provjeriZagrade(char *niz, int *zastavica) {
  if (*niz == '\0') {
    return (*zastavica == 0 ? 1 : 0);
  }
  else {
    if (*niz == '[') ++(*zastavica);
    else --(*zastavica);
    return provjeriZagrade(niz + 1, zastavica);
  }
}

int main() {
  int zastavica = 0;
  if (provjeriZagrade("[[]", &zastavica) == 1) {
    printf("Sve zagrade su uparene");
  }
  else {
    printf("Neke zagrade nisu uparene");
  }
}
```