

Rekurzija - zadatci za vježbu

1. Kvazi-binomni koeficijenti $K(n,m)$ su definirani sa (MI 2013./2014.):

$$K(n, m) = K(n-1, m-1) + m \cdot K(n-1, m+1)$$

$$K(n, n) = K(n, 0) = 1.$$

Ako je $n < 0$ ili $m > n$, onda je $K(n, m) = 0$.

Napišite rekurzivnu funkciju `kvazi_binomni` koja će za bilo koji n i m izračunati vrijednost kvazi-binomnog koeficijenta $K(n,m)$.

2. Zadana je funkcija u C++ kojom se implementira algoritam slijednog traženja. Napisati rekurzivnu verziju funkcije.

```
template <typename T> class LinearSearch : public ISearch<T> {
public:
    RetValSearch search(const T A[], const size_t n, const T &item) override {
        bool found = false;
        int index = -1;
        for (auto i = 0; i < n; i++) {
            if (A[i] == item) {
                found = true;
                index = i;
                break;
            }
        }
        return RetValSearch{found, index};
    }
};
```

3. Zadana je rekurzivna funkcija kojom se implementira kamatni račun. Napisati verziju funkcije korištenjem rekurzije repa (eng. tail recursion).

```
float kamrac(float g, int n, float p) {
    // g - glavnica
    // n - trajanje oročenja u godinama
    // p - kamatna stopa u postotcima
    if (n <= 0)
        return g;
    else
        return (1 + p / 100) * kamrac(g, n - 1, p);
}
```

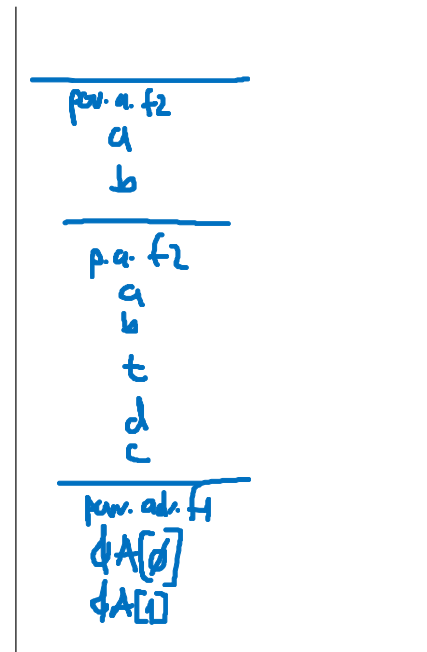
4. Za zadani program prikažite sadržaj sistemskog stoga počevši s pozivom funkcije `f1` iz glavnog programa (linija 20) te sve do neposredno prije izvođenja naredbe `return 0;` u funkciji `f2` (linija 8). Uz svaku stavku napišite broj okteta (byteova) koje stavka zauzima na stogu. Sadržaj stoga te broj okteta koje zauzima svaka stavka prikažite u pravokutniku desno od zadanog programa (**MI 2015./2016.**)

```

1  #include <stdio.h>
2  int f2(int *a, int *b) {
3      if (*a > *b) {
4          --(*a);
5          f2(a, b);
6      }
7      else {
8          return 0;
9      }
10 }
11
12 void f1(int *a, int *b) {
13     int *c = a, *d = b, t;
14     printf("%d %d", *c, *d);
15     t = f2(a, b);
16 }
17
18 int main(void) {
19     int A[] = {4, 3, 2, 1};
20     f1(&A[0], &A[1]);
21     return 0;
22 }

```

Sadržaj stoga:



5. Napisati rekurzivnu funkciju `okreni_oduzmi` koja ispisuje prvih n članova cjelobrojnog polja, gdje se svaki član polja ispisuje umanjen za element na mjestu s indeksom 0. Ispis treba biti u obrnutom redoslijedu, tj. prvo treba ispisati element s indeksom $n-1$. Npr. za polje 2 6 8 11 -3 100 uz ispravne argumente (tj. za $n=5$) ispis prvih 5 članova obrnutim redoslijedom umanjenih za iznos prvog (člana s indeksom 0) je: -5 9 6 4 0 (**MI 2014./2015.**)
6. Napišite rekurzivnu funkciju koja provjerava je li cjelobrojni niz proizvoljne duljine uzlazno sortiran. Složenost funkcije treba biti $\Theta(n)$. Funkcija treba vratiti 1, ako je niz sortiran, a 0 inače. Prototip funkcije je: `int provjeri_sortiranost(int *niz, int n);` Napomena: nerekurzivno rješenje se neće priznati. Prazan niz i niz od samo jednog elementa smatraju se sortiranim. Primjer sortiranih nizova za koje funkcija vraća 1: "1, 2, 3", "1, 2, 2, 3", "1".

Napišite odsječak glavnog programa u kojem se poziva funkcija `provjeri_sortiranost` za cjelobrojni niz "1, 2, 3" (**MI 2017./2018.**).

7. Neka je zadano polje cijelih brojeva A koje ima n elemenata ($A[0], A[1], \dots, A[n-1]$). Potrebno je napisati rekurzivnu funkciju **poljeEkstrema** koja će vratiti 1 ako za svaki $A[i]$ ($i=1, \dots, n-2$), vrijedi ili $A[i-1] < A[i]$ i $A[i] > A[i+1]$ ($A[i]$ je veći od prvih susjeda) ili $A[i-1] > A[i]$ i $A[i] < A[i+1]$ ($A[i]$ je manji od prvih susjeda), a 0 inače. Za $n \leq 2$ funkcija treba vratiti 1. Napomena: nerekurzivno rješenje se neće priznati (**MI 2018./2019.**).
8. Napišite rekurzivnu funkciju koja za znakovni niz (string) proizvoljne duljine provjerava ima li svaka otvorena uglatu zagrada u nizu (znak '[') pripadajuću zatvorenu uglatu zgradu (znak ']') i obratno. Funkcija treba vratiti 1, ako svaka otvorena zagrada u nizu ima pripadajuću zatvorenu zgradu, 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 zgradu. Za niz "[]]" funkcija treba vratiti 0, jer zagrada] nema pripadajuću otvorenu zgradu. Za niz "[[]]" funkcija treba vratiti 1, jer svaka otvorena zagrada ima pripadajuću zatvorenu zgradu.

Napomena: nerekurzivno rješenje se neće priznati.

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

Komentar uz zadatak: uparenost zagrada znači da svaka otvorena uglatu zgradu (znak '[') ima pripadajuću zatvorenu uglatu zgradu (znak ']'), gdje nije bitno da je redoslijed zagrada matematički ispravan, tj. i niz "[]" i niz "[]" su ispravno zadani (**MI 2016./2017.**).

Rješenja:

1.

```
int kvazi_binomni(int n, int m) {
    if ((n < 0) || (m > n))
        return 0;
    if ((m == 0) || (n == m))
        return 1;
    return kvazi_binomni(n - 1, m - 1) + m * kvazi_binomni(n - 1, m + 1);
}
```

2.

```
template <typename T> class LinearSearchRec : public ISearch<T> {
public:
    RetValSearch search(const T A[], const size_t n, const T &item) override {
        if (n == 0) {
            bool found = false;
            int index = -1;
            return RetValSearch{found, index};
        }
        if (A[n - 1] == item) {
            bool found = true;
            int index = n - 1;
            return RetValSearch{found, index};
        }
        return search(A, n - 1, item);
    }
};
```

3.

```
float kamracRec(float g, int n, float p, float rez) {
    if (n <= 0)
        return g * rez;
    float percentage = (1 + p / 100);
    rez = percentage * rez;
    return kamracRec(g, n - 1, p, rez);
}

int main() {
    cout << kamrac(1100, 9, 3);
    cout << kamracRec(1100, 9, 3, 1.0);
    cin.get();
    return 0;
}
```

4.

| | | |
|----------|-----------------|---|
| poziv f2 | Povr. adresa f2 | 4 |
| | a | 4 |
| | b | 4 |
| poziv f2 | Povr. adresa f2 | 4 |
| | a | 4 |
| | b | 4 |
| poziv f1 | t | 4 |
| | d | 4 |
| | c | 4 |
| | Povr. adresa f1 | 4 |
| | &A[0] | 4 |
| | &A[1] | 4 |

5.

```
#include <iostream>
using namespace std;

void okreni_oduzmi(int *a, int m, int n) {
    if (m < n) {
        okreni_oduzmi(a, m + 1, n);
    }
    cout << a[m] - a[0] << " ";
}

int main() {
    int A[] = {2, 6, 8, 11, -3, 100};
    okreni_oduzmi(A, 0, 4);
    cin.get();
    return 0;
}
```

6.

```
#include <iostream>
using namespace std;

int provjeri_sortiranost(int *niz, int n) {
    if (n <= 1)
        return 1;
    if (niz[n - 2] > niz[n - 1])
        return 0;
    return provjeri_sortiranost(niz, n - 1);
}
```

```

int main() {
    int niz1[] = {1, 2, 3};
    std::cout << provjeri_sortiranost(niz1, 3) << std::endl;
    std::cin.get();
    return 0;
}

```

7.

```

int poljeEkstrema(int *a, int n) {
    if (n <= 2) {
        return 1;
    }
    int uv1, uv2;
    if (poljeEkstrema(a, n - 1) == 1) {
        uv1 = (a[n - 3] < a[n - 2]) && (a[n - 2] > a[n - 1]);
        uv2 = (a[n - 3] > a[n - 2]) && (a[n - 2] < a[n - 1]);
        if (uv1 || uv2) {
            return 1;
        }
    }
    return 0;
}

```

8.

```

#include <iostream>
using namespace std;

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) {
        cout << "Sve zagrade su uparene";
    } else {
        cout << "Neke zagrade nisu uparene";
    }
    cin.get();
}

```