****

Tema 7

****

TEME

- Vezbanje takmicarskih zadataka

PRIMER 1: Prilikom sudara dva broja X i Y svaka cifra jednog broja upoređuje se s odgovarajućom cifrom drugog broja, tj jedinca sa jedinicom, desetica sa deseticom, itd. Manja od njih ispada, dok veca ulazi u sastav novoformiranog broja. Ukoliko su jednake, ulaze obe u novonastali broj. Ukoliko u nekom od sudara nema te cifre, uzima se cifra drugog broja.

TEST PRIMERI:

73 28 -> 78
64 357 -> 367
234 135 -> 2335
99099 9999 -> 99999999
2 100 -> 102

POSTUPAK:

- Prebacicemo brojeve u stringove i uporedjivati ih tamo
- Kreiracemo i string kao rezultat
- Posto mogu veliki brojevi uzecemo long long
- V1 ce biti verzija kojom mozemo da uporedjujemo brojeve koji su istociftreni
- V2 ce obraditi slucaj kada se ubaci nula

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

long long a, b;
string A, B, res="";

int main() {
	cin >> a >> b;

	if(b>a) swap(a,b); // hocemo da a bude veci

	A = to_string(a);
	B = to_string(b);

	reverse(A.begin(), A.end());		// obrnemo ih
	reverse(B.begin(), B.end());
```

```
      for(int i=0;i<A.size();i++){    // da bismo isli od i=0 zbog indeksa
         if(A[i]>=B[i]) res+=A[i];
         if(A[i]<=B[i]) res+=B[i];
      }

      reverse(res.begin(),res.end());  // vratimo redosled u pocetni

      cout << res << endl;
      return 0;
}
```

V2

```
for(int i=0;i<A.size();i++){
        if(i>=B.size()){
            res+=A[i];
            continue;
        }
         if(A[i]>=B[i]) res+=A[i];
         if(A[i]<=B[i]) res+=B[i];
}
```

OBJASNJENJE:

- Funkcija swap() se nalazi u iostream
- Funkcija to_string() se nalazi u iostream
- Funkcija reverse() se nalazi u algorithm

PRIMER 2: Program racuna zbir brojeva u datom redu aritmetickog trougla

```
          1
       2  3  4
    5  6  7  8  9
10 11 12 13 14 15 16
...
```

TEST PRIMERI:

2 -> 9
4 -> 91

POSTUPAK:

- Po unosu reda bitno je da identifikujemo koji je pocetni broj u opsegu koji treba da se sabira, a pocetni broj u redu se dobija kada se na prvi broj u prethodnom redu doda broj elemenata u tom redu
- U svakom redu se povecava po 2 elementa koji trebaju da se saberu
- Nakon toga, racunamo koliko se brojeva treba da sabere
- Pravimo algoritam za sabiranje

```
#include <iostream>
using namespace std;
int main() {
     long long k;
     cin >> k;

     // odredjujemo prvi broj u k-tom redu trougla
     long long pocetak = 1;
     long long brojElemenata = 1;
     for (int i = 1; i < k; i++) {
          pocetak += brojElemenata;
          brojElemenata += 2;
     }
     // odredjujemo zbir elemenata u k-tom redu trougla
     long long zbirRedaTrougla = 0;
     for (long long i = pocetak; i < pocetak + brojElemenata; i++)
          zbirRedaTrougla += i;
     cout << zbirRedaTrougla << endl;
     return 0;
}
```

RESENJE SA REKURZIJOM

```
#include <iostream>
using namespace std;
/*
                1
           2    3    4
      5    6    7    8    9
10   11   12   13   14   15   16
…

*/
long long brElemenata(long long red) {
    if (red > 1) {
            return 2 + brElemenata(red-1);
    }
    else {
            return 1;
    }
}

long long pocetniBroj(long long red) {
    if (red > 1) {
            return (pocetniBroj(red-1) + brElemenata(red-1));
    } else {
        return 1;
    }
```

```
}

long long zbir(long long red) {
    long long p = pocetniBroj(red);
    long long brEl = brElemenata(red);
    long long zbirBr = 0;
    for (long long i = p; i < p+brEl; i++) {
        zbirBr += i;
    }
    return zbirBr;
}

int main() {
    long long k;
    cin >> k;
    cout << zbir(k) << endl;
    return 0;
    }
```

OBJASNJENJE :

- Bitno skrenuti paznju pri pozivu rekurzije da arg mora biti drugaciji od inicijalnog
- Rekurzija mora imati siguran izlaz
- Rekurzija se mora konstruisati na optimalan nacin (da se prvo radi slucaj koji se cesce desava - da je broj > 1, od onog sto se desava samo 1 - da je broj == 1)

PRIMER 3: Pokrenuti progtak arena okruzenje sa petlje, kao i kako izgledaju zadaci.