Recapitulare functii recursive

Parte teoretica

- In termeni simpli, o functie recursiva este o functie in care apare un apel catre ea insasi!
- Cand scriem o functie recursiva, trebuie sa avem in vedere urmatoarele 2 lucruri:
  - Logica de oprire
  - Apelul recursiv trebuie facut pe un set de date mai mic decat apelul anterior.
- In cazul in care, nu avem grija la cele doua puncte importante de mai sus, functia noastra nu are cum sa fie valida din punct de vedere al logicii, avand sanse mari ca rezultatul sa nu fie cel dorit.

Probleme des intalnite

Mai jos o sa vedem cateva din problemele a caror rezolvare este facuta folosind recursivitate. Pentru fiecare din ele, o sa vedem mai intai varianta simpla si dupa aceea pe cea recursiva, pentru a le putea compara.

1. Suma primelor n numere
   - Varianta fara recursivitate

```
    int calculareSumaPrimelorN(int n){
        int sum = 0;
        for (int i = 0; i <= n; i++) {
            sum += i;
        }
        return sum;
    }
```

   - Varianta cu recursivitate

```
int calculareSumaPrimelorNRecursiv(int n) {
    if (n == 0) {
        return 0;
    }
    return n + calculareSumaPrimelorNRecursiv(n - 1);
}
```

   - Se poate observa cum in cazul functiei recursive, dupa fiecare apel, `n` va avea o valoare cu 1 mai mica, lucru ce va apropria apelul functiei de punctul de oprire.

2. Calculare `n` factorial:
   - Varianta fara recursivitate

```

```

```
    int factorial(int n) {
        int rezultat = 1;
        for (int i = 1; i <= n; i++) {
            rezultat *= i;
        }
        return rezultat;
    }
        ```
```

- Varianta cu recursivitate

```
    int factorialRecursiv(int n) {
        if (n <= 1) {
            return 1;
        }
        return n * factorialRecursiv(n-1);
    }
```

3. Afisarea celui de-al n numar Fibonacci

- Varianta fara recursivitate

```
int fibonacci(int n) {
    if (n <= 2) {
        return 1;
    }
    int t0 = 1;
    int t1 = 1;
    int t2;
    for (int i = 2; i < n; i++) {
        t2 = t0+t1;
        t0 = t1;
        t1 = t2;
    }

    return t2;
}
```

- Varianta recursiva

```
int fibonacciRecursive(int n) {
    if(n == 0){
        return 0;
    } else if(n == 1) {
        return 1;
    } else {
```
```

```
        return (fibonacciRecursive(n-1) + fibonacciRecursive(n-
2));
    }
    }
```

IMPORTANT:

- Orice functie recursiva se poate scrie sub forma unei instructiuni repetitive
- Atunci cand incercati sa evaluati o functie recursiva, va puteti gandii ca fiecare apel intoarce un rezultat temporar iar atunci cand atinge punctul de oprire, aceste rezultate intermediare sunt colectate.

Exercitii cu recursivitate Bacalaureat

1. Subprogramul f este definit alăturat. Indicați valoarea `f(4770777,7)`.

```
int f (int n, int k)
{ if (n!=0)
if(n%10==k) return 1+f(n/10,k);
return 0;
}
```

   - Raspuns corect: 3

2. Subprogramul f este definit alăturat. Indicați ce se afișează în urma apelului de mai jos: `f(4)`

```
void f(int x)
{ while(x>1){ x=x-1; f(x-1);}
cout<<x; | printf(”%d”,x);
}
```

   - Raspuns corect: `01101`
   - Atentie aici la apelurile imbricate din while.

3. Subprogramul f este definit alăturat. Indicați ce se afișează în urma apelului de mai jos: `f(54321)`

```
void f (int x)
{ cout<<"*"; | printf("*");
   if(x>0)
   { cout<<x; | printf("%d",x);
    f(x/100);
   }
cout<<"/"; | printf("/");
}
```

- Raspuns corect: `*54321*543*5*////`
- Atentie la faptul ca pentru fiecare apel, va trebui sa afisam `/`

4. Subprogramul f este definit alăturat. Indicați valoarea `f(38627)`

```
int f(int n)
{ int c;
 if (n==0) return 9;
 c=f(n/10); if (n%10<c) return n%10;
 return c;
}
```

- Raspuns corect: `2`
- Atentie mare la ce se intoarce si valoarea lui `n` la fiecare pas.

5. Subprogramul f este definit alăturat. Indicați ce se afișează în urma apelului de mai jos. f(7552021,1);

```
void f (int n, int k)
{ if (n!=0)
  { f(n/10,k+1);
    if(n%10==k) cout<<k; | printf("%d",k);
  }
}
```

- Raspuns corect: `7521`

6. Subprogramul f este definit alăturat. Indicaţi valoarea lui `f(2121,19).`

```
int f(int x, int y)
{ if(x==0) return 0;
  if(y==0) return 1;
  return x%2+y%2+f(x/10, y/10);
}
```

- Raspuns corect: `4`

7. Subprogramele `f1` și `f2` sunt definite mai jos. Indicați valoarea `f2(41382)`.

```
int f1(int c)
{ if (c%2==1) return 1;
  else return 2;
}

int f2(int n)
{ if (n==0) return 0;
```

```
  else return f1(n%10)+f2(n/10);
}
```

- Raspuns corect: 8

8. Se consideră subprogramele f și g definite mai jos.

```
int g(int x)
{ if (x>9) return (x/10 + x%10);
  return x;
}

int f(int c)
{ if (c<1) return 1;
  return g(c+f(c-1));
}
```

Indicați o mulțime de valori posibile pentru variabila întreagă a, astfel încât, pentru fiecare dintre acestea, valoarea f(a) să fie egală cu 2.:

- {4, 6}
- {7, 9}
- {1,3,8}
- {1,4,7} (RASPUNS CORECT)

9. Subprogramul f este definit alăturat. Scrieți toate valorile naturale din intervalul [1,10] pe care le poate avea x, astfel încât valoarea lui f(10,x) să fie un număr strict mai mare decât 20.

```
int f(int a,int b)
{ if(a>b) return a/b+f(a-b,b);
  if(a<b) return b/a+f(a,b-a);
  return 1;
}
```

- Raspuns corect: 1 si 9.
- Acest exercitiu presupune efectuarea tuturor calculelor pentru a afla raspunsul corect.