Temă pentru acasă. Transformări de tablouri

1. Implementaţi, corespunzător cerinţelor fiecărui caz în parte, funcţia

```
void transformaPeLoc(int a[], int n){ ... }
```

care găseşte la intrarea în execuţie, în tabloul `a`, valorile iniţiale a_0, a_1, ..., a_{n-1} pe care le transformă în valorile finale $\tilde{a}_0$, $\tilde{a}_1$, ..., $\tilde{a}_{n-1}$.

Atenţie, deoarece nu este precizată o valoare maximă pentru numărul n de elemente, în corpul funcţiei nu pot fi declarate tablouri auxiliare de dimensiuni comparabile cu n şi, prin urmare, calculele trebuie efectuate pe loc în tabloul `a`.

i) $\tilde{a}_i = a_{n-1-i}$, $i = \overline{0, n-1}$.

ii) $\tilde{a}_i = a_i - a_{n-1-i}$, $i = \overline{0, n-1}$.

iii) $\tilde{a}_i = a_0 a_i^2 + a_1 a_{i-1}^2 + \cdots + a_i a_0^2$, $i = \overline{0, n-1}$.

iv) $\tilde{a}_0 = a_0$, şi $\tilde{a}_i = a_{i-1} + a_i$, $i = \overline{1, n-1}$.

v) $\tilde{a}_0 = a_0$, $\tilde{a}_{n-1} = a_{n-1}$ şi $\tilde{a}_i = a_{i-1} + a_{i+1}$, $i = \overline{1, n-2}$.

vi) $\tilde{a}_0 = a_0$, $\tilde{a}_{n-1} = a_{n-1}$ şi $\tilde{a}_i = a_{i-1} - \tilde{a}_{i+1}$, $i = \overline{1, n-2}$.

vii) $\tilde{a}_i = a_0 + a_1 + \ldots + a_i - a_{i+1} - a_{i+2} - \ldots - a_{n-1}$, $i = \overline{0, n-1}$.

viii) $\tilde{a}_i = a_0 + a_1 + \ldots + a_i - \tilde{a}_{i+1} - \tilde{a}_{i+2} - \ldots - \tilde{a}_{n-1}$, $i = \overline{0, n-1}$.

ix) $\tilde{a}_i = \min\{a_0, a_1, \ldots, a_i\}$, $i = \overline{0, n-1}$.

x) $\tilde{a}_i = \min\{a_0, a_1, \ldots, a_i, -a_{i+1}, -a_{i+2}, \ldots, -a_{n-1}\}$, $i = \overline{0, n-1}$.

xi) $\tilde{a}_i = \min\{a_0, a_1, \ldots, a_i, -\tilde{a}_{i+1}, -\tilde{a}_{i+2}, \ldots, -\tilde{a}_{n-1}\}$, $i = \overline{0, n-1}$.

xii) $\tilde{a}_i = \min\{a_0 a_{n-1}, a_1 a_{n-2}, \ldots, a_i a_{n-1-i}\}$, $i = \overline{0, n-1}$.

2. Definiţi, corespunzător cerinţelor fiecărui caz în parte, funcţia

```
void transformaAinB(int a[],  int b[], int n){ ... }
```

care calculează valorile finale $\tilde{b}_0$, $\tilde{b}_1$, ..., $\tilde{b}_{n-1}$ pe baza valorilor iniţiale a_0, a_1, ..., a_{n-1} şi b_0, b_1, ..., b_{n-1} aflate în tablourile `a` şi `b`.

La apelare cele două tablouri sunt presupuse disjuncte, în urma apelului valorile lui `a` sunt nedefinite.

i) $\tilde{b}_i = a_{n-1} a_0^2 + a_{n-2} a_1^2 + \cdots + a_{n-1-i} a_i^2$, $i = \overline{0, n-1}$.

ii) $\tilde{b}_i = a_{n-1} b_0^2 + a_{n-2} b_1^2 + \cdots + a_{n-1-i} b_i^2$, $i = \overline{0, n-1}$.

iii) $\tilde{b}_i = b_{n-1}a_0^2 + b_{n-2}a_1^2 + \cdots + b_{n-1-i}a_i^2$, $i = \overline{0, n-1}$.

3. Implementaţi funcţia

```
void mutaPareImpare(int a[], int dim){ ... }
```

care schimbă locurile elementelor tabloului a corespunzător cerinţelor fiecărui caz în parte. Mutarea trebuie să păstreze ordinea locurilor în tablou pentru elementele pare între ele şi a celor impare între ele (nici un element nu poate sări peste altul de aceeaşi paritate)

i) In tabloul final toate elementele pare sunt în faţă iar cele impare în spate.

ii) Primul element va fi lăsat pe loc, după care paritatea elementelor alternează până se epuizează una dintre parităţi.

Exemplu de rezolvare:

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

void mutaPareImpare0(int a[], int dim){
    int aux[100];//cu ajutor
    int pare=0;
    for(int i=0;i<dim;i++){
        if(a[i]%2==0) aux[pare++]=a[i];
    }
    int impare=dim-1;
    for(int i=dim-1; i>=0;i--){
        if(a[i]%2!=0) aux[impare--]=a[i];
    }
    for(int i=0;i<dim;i++) {
        cout<<(a[i]=aux[i])<<" ";
    }
    cout<<endl;
}

void mutaPareImpare(int a[], int dim){
    int i=0;
    while(i<dim){
        //a[0],...,a[i-1] sunt pare
        int val=a[i];
        if(val%2==0) {
            i++;
            continue;
        }
        //val este impar
        //cautam primul par dupa el
        int j=i+1;
        for(;j<dim;j++){
```

```
            if(a[j]%2==0) break;
        }
        if(j==dim) break;//toate dupa val sunt impare
        //schimbam ciclic a[j] cu a[i]
        a[i]=a[j];
        for(int h=j-1;h>=i+1;h--){
            a[h+1]=a[h];
        }
        a[i+1]=val;
    }

}
int main(){
    const int dim =10;
    int tab[dim]={1, 2, -30, -41, -50, 63, 6, 61, -90, 10};
    mutaPareImpare(tab,dim);
    for(int i=0;i<dim;i++) {
        cout<<tab[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```