Fundamentele programarii

Temă pentru acasă. Funcţii de decizie

Completaţi definiţiile următoarelor funcţii de decizie care returnează `true` sau `false` după cum găsesc sau nu ceea ce caută. Presupunem că este definită constanta globală `dimMax=100` iar dimensiunea actuală a tablourilor este dată de parametrul formal `n`.

1.) `bool are1Zero(int tab[dimMax], int n){...}` decide dacă în tabloul `tab` există elemente nule;

2.) `bool are2Egale(int tab[dimMax], int n){...}` decide dacă în `tab` există două elemente distincte având aceeaşi valoare;

3.) `bool are3Pare(int tab[dimMax], int n){...}` decide dacă în tabloul `tab` există măcar trei elemente pare;

4.) `bool areUnParMare(int tab[dimMax], int n){...}` decide dacă tabloul `tab` are un element par strict mai mare decât toate elementele impare;

5.) `bool estePozitiv(int tab[dimMax], int n){...}` decide dacă toate elementele tabloului `tab` sunt strict pozitive;

6.) `bool estePestrit(int tab[dimMax], int n){...}` decide dacă în `tab` elementele pare şi elementele impare sunt dispuse alternant;

7.) `bool esteEchilibrat(int tab[dimMax], int n){...}` decide dacă există un indice i astfel încât suma elementelor până la i inclusiv este egală cu suma celor de după i;

8.) `bool esteCrescator(int tab[dimMax], int n){...}` decide dacă tabloul `tab` este ordonat crescător (nestrict);

9.) `bool esteOscilant(int tab[dimMax], int n){...}` decide dacă în `tab` creşterile (`tab[i]<=tab[i+1]`) şi descreşterile (`tab[i]>=tab[i+1]`) alternează;

10.) `bool esteConvex(int tab[dimMax], int n){...}` decide dacă, la parcurgerea tabloului `tab`, valorile mai întâi descresc până ajung la un minim, după care cresc;

11.) `bool esteSimetrica(int A[dimMax][dimMax], int n){...}` decide dacă matricea pătratică $A = (a_{ij})$ este simetrică (adică $a_{ij} = a_{j\,i}\, \forall i, j \in \{0, 1, \ldots, n-1\}$);

12.) `bool esteDiagDominanta(int A[dimMax][dimMax], int n){...}` decide dacă fiecare element de pe diagonala principală a matricei pătratice $A = (a_{ij})$ este mai mare în modul decât suma modulelor celorlalte elemente de pe linia sa, adică dacă pentru orice $i \in \{0, 1, \ldots, n-1\}$ avem

$$|a_{ii}| \geq |a_{i0}| + \cdots + |a_{ii-1}| + |a_{ii+1}| + \cdots + |a_{in-1}|.$$

13.) `bool areLiniePozitiva(int A[dimMax][dimMax], int n){...}` decide dacă în matricea `A` există o linie cu toate elementele strict pozitive;

14.) `bool areVarf(int A[dimMax][dimMax], int n){...}` decide dacă în matricea `A` există un element nenul pentru care toate celelalte elemente de pe linia şi coloana sa sunt nule;

15.) `bool arePunctSa(int A[dimMax][dimMax], int n){...}` decide dacă matricea `A` are un punct şa, adică un element care este în acelaşi timp cel mai mic de pe linia sa şi cel mai mare de pe coloana sa;

Exemplu de rezolvare:

```
#include<iostream>
using namespace std;
const int dimMax = 100;
bool estePozitiv1(int tab[dimMax], int n){
    //cautam un contraexemplu
    for (int i = 0; i<n; i++){
        if (tab[i] <= 0) return false;
    }
    return true;
}
bool estePozitiv2(int tab[dimMax], int n){
    //cautam un contraexemplu
    bool amGasitUnNegativ = false;
    for (int i = 0; i<n && !amGasitUnNegativ; i++){
    }
    return !amGasitUnNegativ;
}
bool estePozitiv3(int tab[dimMax], int n){
    int i; // i arata pana unde sunt numai elem. strict pozitive
    for (i = 0; i<n && tab[i]>0; i++)
        ;
    return i == n ? true : false;
}

int main(void){
    int vect[dimMax] = { 1, 2, 6, 41, 10, 8, 10, 30, 50, 71 };
    if (estePozitiv1(vect, 10)) cout << "DA "; else cout << "NU ";
    if (estePozitiv2(vect, 10)) cout << "DA "; else cout << "NU ";
    if (estePozitiv3(vect, 10)) cout << "DA "; else cout << "NU ";
    cout << endl;
    return 0;
}
```