**Exerciții**

1. (0,05p fiecare)  
   Scrieti functii cu numele f1, f2, f3 si definite corespunzator (de ex. int f1(int x[100], int n) {…} )pentru a calcula urmatoarele expresii:  
   a) f1= 1\*x[1]+2\*x[2]+3\*x[3]+…+n\*x[n];  
   b) f2 = 1\*x[n]+2\*x[n-1]+3\*x[n-2]+…+2\*x[n-1]+1\*x[n];  
   c) f3 = x[1]+x[1]\*x[2]+x[1]\*x[2]\*x[3]+…+x[1]\*x[2]\*…\*x[n];
2. (0,05p fiecare) Scrieti functii cu numele f4, f5, f6, si definite corespunzator (de ex. int f4(int x[100], int y[100], int n) {…} )pentru a calcula urmatoarele expresii:  
   a) f4= min(x[1],y[1])+min(x[2],y[2])+…+min(x[n],y[n]);  
   b) f5 = x[1]/y[n]-x[2]/y[n-1]+x[3]/y[n-2]-…+…-…….. x[n]/y[1];  
   c) f6 = s(x[1]+y[1])+s(x[2]+y[2])+…+s(x[n]+y[n]), unde s(a) este suma cifrelor numarului a;
3. (0,05 fiecare) Scrieti functii cu numele f7, f8, f9 si definite corespunzator (de ex. void f7(int x[100], int y[100], int z[100], int n) {…} )pentru a calcula valorile vectorului z in functie de valorile vectorilor x si y, astfel:  
   a) z[1]= min(x[1],y[1]) si z[2]=min(x[2],y[2]) si … si z[n]=min(x[n],y[n]);  
   b) z[1] =cmmdc(x[1],y[n]) si z[2]=cmmdc(x[2],y[n-1]) si … si z[n]=cmmdc(x[n],y[1]), unde cmmdc(a,b) este cel mai mare divizor comun al numerelor a si b;  
   c) z[1] = uc(x[1]+y[1]), z[2]=uc(x[2]+y[2]),…, z[n]=uc(x[n],y[n]), unde uc(a) este ultima cifra a numarului a;
4. (0,05 fiecare) Scrieti functii cu numele f10, f11, f12 si definite corespunzator (de ex. bool f10(int x[100], int y[100], int z[100], int n) {…} ) pentru a testa daca valorile vectorului z verifica anumite conditii, in functie de valorile vectorilor x si y, astfel:  
   a) z[1]==x[1]\*y[1] si z[2]==x[2]\*y[2] si … si z[n]==x[n]\*y[n];  
   b) z[1]==cmmmc(x[1]\*2,y[n]\*3) si z[2]=cmmmc(x[2]\*2,y[n-1]\*3) si … si z[n]==cmmmc(x[n]\*2,y[1]\*3), unde cmmmc(a,b) este cel mai mic multiplu comun al numerelor a si b;  
   c) z[1]==pc(x[1]+y[1]\*1) si z[2]==pc(x[2]+y[2]\*2) si … si z[n]==pc(x[n]+y[n]\*n), unde pc(a) este prima cifra a numarului a;
5. (0,05 fiecare) Scrieti functii cu numele f13, f14, f15 si definite corespunzator (de ex. int f15(int x[100], int n) {…} ) pentru a returna valoarea k1, k2, k3, care reprezinta:  
   a) k1 = 1, daca toate elementele din x sunt identice intre ele, respectiv 0 in caz contrar;  
   b) k2 = numarul de perechi de elemente prime intre ele, aflate pe pozitii egal distantate de centrul vectorului;  
   c) k3 = numarul de aparitii ale valorii x[1] in vectorul x;
6. (0,05 fiecare) Scrieti functii cu numele f16, f17, f18 si definite corespunzator (de ex. void f16(int x[100], int n) {…} ) pentru a modifica vectorul x, astfel incat elementele acestuia sa ajunga in ordinea urmatoare:  
   a) x[n], x[n-1], x[n-2], …, x[3], x[2], x[1];  
   b) x[2], x[1], x[4], x[3], …., x[n], x[n-1]; (aici n este par)  
   c) x[2], x[3], x[4], …, x[n], x[1];
7. (0,05 fiecare) Scrieti functii cu numele f19, f20, f21 si definite corespunzator (de ex. void f19(int x[100], int n) {…} ) pentru a modifica vectorul x, astfel incat elementele acestuia sa ajunga in ordinea urmatoare (sau sa aiba valorile urmatoare):  
   a) x[n],x[1],x[2],…,x[n-1];  
   b) x[1], 0, x[3], 0, x[5], 0, …;  
   c) 1\*x[1], 2\*x[2], 3\*x[3], …, n\*x[n];
8. (0,05 fiecare) Scrieti functii cu numele f22, f23, f24 si definite corespunzator (de ex. void f23(int x[100], int n) {…} ) pentru a modifica vectorul x, astfel incat elementele acestuia sa ajunga in ordinea urmatoare (sau sa abia valorile urmatoare):  
   a) x[1], -x[2], x[3], -x[4], …, ;  
   b) 0, x[2], x[3], 0, x[5], x[6], 0, x[8], x[9], …;  
   c) n\*x[1], (n-1)\*x[2], (n-2)\*x[3], …, 1\*x[n];
9. (0,05 fiecare) Scrieti functii cu numele f25, f26, f27 si definite corespunzator (de ex. int f25(int x[100], int n) {…} ) pentru a calcula valoarea k4, k5, k6, care reprezinta:  
   a) k4 = numarul de elemente din x care sunt egale cu suma vecinilor sai;  
   b) k5 = numarul de perechi de elemente prime intre ele, aflate pe pozitii consecutive in vector;  
   c) k6 = numarul de elemente negative din vector, aflate pe pozitii divizibile prin 3;
10. (0,05 fiecare) Scrieti functii cu numele f28, f29, f30 si definite corespunzator (de ex. int f28(int x[100], int n) {…} ) pentru a calcula valoarea k7, k8, k9, care reprezinta:  
    a) k7 = numarul de elemente din x care sunt mai mari decat media aritmetica a elementelor vectorului;  
    b) k8 = numarul de elemente din vector care sunt egale cu oglinditele lor;  
    c) k9 = numarul de elemente pozitive si pare din vector, aflate pe pozitii impare;
11. (1,5p) Având un vector cu numere intregi sortate in ordine crescatoare, creati un arbore binar de cautare, balansat, bazat pe acest vector.  
    struct TreeNode {  
    int val;  
    TreeNode \*left;  
    TreeNode \*right;  
    TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
    };  
    TreeNode\* sortedArrayToBST(int nums[]) {

};

1. (1,00p) Având un arbore binar, calculati inaltimea acestuia. Inaltimea este numarul de noduri din cel mai lung drum de la nodul root pana la un nod frunza.  
   struct TreeNode {  
   int val;  
   TreeNode \*left;  
   TreeNode \*right;  
   TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
   };  
   int maxDepth(TreeNode\* root) {}
2. (1,00p) Având un arbore binar, calculati inaltimea minima a acestuia. Inaltimea minima este numarul de noduri din cel mai scurt drum de la nodul root pana la un nod frunza.  
   struct TreeNode {  
   int val;  
   TreeNode \*left;  
   TreeNode \*right;  
   TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
   };  
   int minDepth(TreeNode\* root) {}
3. (1,00p) Având un arbore binar, si o valoare sum, determinati daca exista un drum de la nodul root pana la orice nod frunza, astfel incat suma valorilor nodurilor sa fie egala cu sum.  
   struct TreeNode {  
   int val;  
   TreeNode \*left;  
   TreeNode \*right;  
   TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
   };  
   bool hasPathSum(TreeNode\* root, int sum) {}
4. **Fiind dat urmatorul arbore binar, si *sum = 22*,**
5. 5
6. / \
7. 4 8
8. / / \
9. 11 13 4
10. / \ \
11. 7 2 1

**returnati true, deoarece exista un drum de la root la o frunza 5->4->11->2 care are suma = 22.**

1. (1,00p) Implementati un program care inverseaza un arbore binar.  
   **Exemplu:**  
   Din
2. 4
3. / \
4. 2 7
5. / \ / \
6. 1 3 6 9

in

4

/ \

7 2

/ \ / \

9 6 3 1

TreeNode\* invertTree(TreeNode\* root) {

}

1. (1,50p) Avand un arbore binar, returnati media nodurilor de pe fiecare nivel, sub forma de vector.
2. Input:
3. 3
4. / \
5. 9 20
6. / \
7. 15 7
8. Output: [3, 14.5, 11]

Explicatie:  
Media nodurilor de pe nivelul 0 este 3, pe nivelul 1 este 14.5, si pe nivelul 2 este 11. Deci returnam [3, 14.5, 11].  
  
double[] averageOfLevels(TreeNode\* root) {  
}

1. (1,50p) Construiti reprezentarea String al unui arbore binar.
2. Exemplul 1:
3. Input:
4. 1
5. / \
6. 2 3
7. /
8. 4
9. Output: "1(2(4))(3)"

Explicatie: Initial, raspunsul ar trebui sa fie „1(2(4)())(3()())”, dar trebuie sa evitati toate perechile de paranteze care nu sunt necesare(perechi de paranteze goale).  
Raspunsul devine „1(2(4))(3)”.

Exemplul 2:

Input:

1

/ \

2 3

\

4

Output: "1(2()(4))(3)"

char\* tree2str(TreeNode\* t) {  
}