Pe prima pagină de examen scrieți în colțul dreapta sus
Numele și prenumele

Examen PC – 29 ianuarie 2024

1p din oficiu

1. (2p) Scrieți în ordine, pe 8 rânduri separate, rezultatele afișate la ieșirea standard obținute în urma execuției programului C de mai jos:

```c
#include <stdio.h>
#include <string.h>

#define RAPORT(x,y) (x)/y

void dubleaza(double x[])
{
    static int i=1;
    x = x+i;
    x[0] *= 2;
    i++;
}

int main()
{
    int a=21;
    int b=10;
    printf("%d\n", RAPORT(b-a,b+a));

    float f = 1.0f;
    for (int i=0; i<1000; i++)
    {
        if (i>100)
            break;
        f *= 10.0f;
    }
    printf("%f\n",f);

    int c = a & b;
    printf("%d\n", c);

    double t[] = {9.9, 7.7, 5.5, 3.3, 1.1};
    printf("%x\n", sizeof(t));

    dubleaza(t); dubleaza(t); dubleaza(t);
    printf("%f\n", t[0]+t[1]+t[2]+t[3]+t[4]);

    char s[10]="examen";
    char r[10]="examen";
    printf("%s\n", (s!=r)?"POINTER":"VALOARE");

    enum {zece=10, douazeci=20, treizeci};
    printf("%d\n", zece+douazeci+treizeci);

    struct data {
        char nume[4];
        int valoare;
    }p;

    p.valoare=400;
    strcpy(p.nume,"Popa");
    printf("%d\n",p.valoare);

    return 0;
}
```

2. (2p) Scrieți în C o funcție numită *timp_ramas* care primește ca și parametru un moment de timp dintr-o zi și returnează sub forma unui tablou cu trei elemente (ore, minute, secunde) timpul rămas din ziua respectivă. Scrieți apoi un program care citește de la intrarea standard un număr natural pozitiv *n* (mai mic decât 1000) și apoi *n* momente de timp de forma **HH:MM:SS** (**HH** orele 0...23, **MM** minutele 0...59, **SS** secundele 0...59) și care utilizează funcția *timp_ramas* pentru a afișa pe câte un rând separat timpul rămas (**HH:MM:SS**) de la fiecare moment de timp citit până la sfârșitul zilei. În cazul în care datele de intrare nu respectă specificația se va afișa un mesaj de eroare.

Exemple:

Citire de la intrarea standard:	Afișare la ieșirea standard:
3 10:57:35 20:00:00 18:24:58	13:02:25 04:00:00 05:35:02
2 31:45:20 17:60:02	Date de intrare invalide!

3. (2p) Scrieți în C o funcție recursivă numită *identice* care primește ca și parametru un număr întreg pozitiv din intervalul 10^1...10^9 și care returnează 1 dacă acesta are toate cifrele identice, iar 0 în caz contrar. În cazul în care numărul nu respectă specificația, funcția va returna valoarea -1.

Exemple:

Înainte de apel:	Apelul funcției:	După apelul funcției:
a←8888888	x←identice(a)	x→1
a←55555	x←identice(a)	x→1
a←77377777	x←identice(a)	x→0
a←99	x←identice(a)	x→1
a←4	x←identice(a)	x→-1
a←-3333	x←identice(a)	x→-1

4. (3p) În fișierul text *numere.txt* sunt scrise pe fiecare linie câte un număr de înmatriculare al unui autoturism din România. Numerele sunt de forma **JJNNAAA** sau **JNNNAAA** sau **JNNAAA**, unde **J** este o literă componentă a numelui județului, **N** o cifră, **A** o literă. Scrieți un program C, structurat în funcții, care citește datele din fișierul de intrare și care determină județul/județele în care sunt înmatriculate cele mai multe autoturisme. Rezultatul se va scrie într-un fișier text, al cărui nume este trimis ca și prim argument la execuția programului. Se presupune că datele existente în fișierul de intrare sunt valide. Afișați câte un mesaj de eroare în cazurile când nu se poate citi fișierul de intrare și în cazul în care programul este executat fără argumente.

Exemplu:

Conținutul fișierului text de intrare *numere.txt*:	Rezultatul scris în fișierul text de ieșire:
CJ04ART CJ01ABC AB48BAM B87GMB B345PKM B21WSD CJ03RFD MS42EDF B555WWW SB21MBV CJ99HHG	Judetul/Judetele cu cele mai multe autoturisme: CJ B