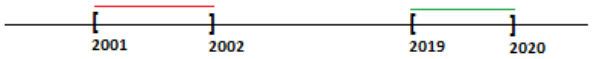
1. Expresia C/C++
`!(x<2001) && (x<=2002) || (x>=2019) && !(x>2020)`
are valoarea 1 dacă și numai dacă valoarea memorată de variabila reală **x** aparține reuniunii:

a. {2001} ∪ [2002,2019] ∪ {2020}
b. [2001,2002] ∪ {2019} ∪ {2020}
c. [2001,2002] ∪ {2019,2020}
d. [2001,2002] ∪ [2019,2020]



`!(x<2001) && (x<=2002) || (x>=2019) && !(x>2020)`

2. Subprogramul `f` este definit alăturat. Indicați ce se afișează în urma apelului de mai jos.
`f(12345);`

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

a. +++++1+123+12345+
b. ++++112312345
c. +12345+123+1+++++
d. +1+123+12345+

x=12345	**Stiva:**
scrie +	12345
x>0 (A) => f(123)	12345 123
f(123)	
scrie +	
x>0 (A) => f(1)	12345 123 1
scrie +	
x>0 (A) => f(0)	12345 123 1 0
scrie +	
x>0 (F)	
scrie +	12345 123 1
scrie 1	
scrie +	12345 123
scrie 123	
scrie +	1234
scrie 12345	
scrie +	Nu mai sunt elemente in stiva

3. Utilizând metoda backtracking, se generează toate modalitățile de forma un grup de patru arbori cu flori din mulțimea {**albizia**, **jacaranda**, **laburnum**, **magnolie**, **mimoza**, **sakura**}. Două grupuri diferă prin cel puțin un arbore. Primele cinci soluții generate sunt, în această ordine: **(albizia, jacaranda, laburnum, magnolie)**, **(albizia, jacaranda, laburnum, mimoza)**, **(albizia, jacaranda, laburnum, sakura)**, **(albizia, jacaranda, magnolie, mimoza)**, **(albizia, jacaranda, magnolie, sakura)**. Indicați o enumerare care este generată ca soluție.

a. (magnolie, mimoza,laburnum, sakura)
b. (jacaranda, magnolie, mimoza, sakura)
c. (jacaranda, laburnum, magnolie, mimoza, sakura)
d. (albizia, jacaranda, magnolie, laburnum)

4. Un graf orientat cu 5 vârfuri este reprezentat prin matricea de adiacență alăturată. Indicați numărul vârfurilor cu gradul interior 2.

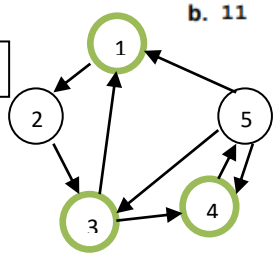
```
0 1 0 0 0
0 0 1 0 0
1 0 0 1 0
0 0 0 0 1
1 0 1 1 0
```

a. 1
b. 2
c. 3
d. 4

5. Un graf neorientat are 20 de noduri și 5 componente conexe, fiecare dintre acestea fiind arbore. Indicați numărul de muchii ale grafului.

a. 7
b. 11
c. 15
d. 19

I.4



I.3. Indiferent cum alegem sa formam 5 subgrafuri care sa indeplineasca conditia de arbore, numarul total de muchii este 15.
De exemplu, sa presupunem ca avem o padure formata din 5 arbori, fiecare arbore avand cate 4 noduri; prin urmare, fiecare arbore din padure are 3 muchii; numarul total de muchii este 3*5=15

SUBIECTUL al II-lea **(40 de puncte)**

1. **Algoritmul alăturat este reprezentat în pseudocod.**
S-a notat cu `a%b` restul împărțirii numărului natural `a` la numărul natural nenul `b` și cu `[c]` partea întreagă a numărului real `c`.

```
citește n
  (număr natural nenul)
x←1; m←0; p←1
┌cât timp x<10 execută
| cn←n
| ┌cât timp cn≠0 execută
| | c←cn%10; cn←[cn/10]
| | ┌dacă c=x atunci
| | | m←c*p+m; p←p*10
| | └■
| └■
| x←x+2
└■
scrie m
```

a. Scrieți ce se afișează dacă se citește numărul `253387`. **(6p.)**
7533

b. Scrieți cel mai mic și cel mai mare număr din intervalul $[10,10^2)$ care pot fi citite astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului, să se afișeze `0`. **(6p.)**
20 88

c. Scrieți programul C/C++ corespunzător algoritmului dat. **(10p.)**

d. Scrieți în pseudocod un algoritm echivalent cu cel dat, înlocuind adecvat prima structură repetitivă cu o structură repetitivă de alt tip. **(6p.)**

II.1.c

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

unsigned long int n,cn,p,m;
unsigned int x,c;
int main()
{
   cin>>n;
   x=1;m=0;p=1;
   while(x<10)
   {
      cn=n;
      while(cn!=0)
      {
         c=cn%10;cn=cn/10;
         if(c==x)
         {
            m=c*p+m;p=p*10;
         }
      }
      x=x+2;
   }
   cout<<m;
   return 0;
}
```

II.1.d

```
citește n
  (număr natural nenul)
x←1; m←0; p←1
┌cât timp x<10 execută
| cn←n
| ┌cât timp cn≠0 execută
| | c←cn%10; cn←[cn/10]
| | ┌dacă c=x atunci
| | | m←c*p+m; p←p*10
| | └■
| └■
| x←x+2
└■
scrie m
```

```
citeste n
x=1;m=0;p=1;
daca(x<10)
repeta
   cn=n;
   cat timp(cn!=0) executa
      c=cn%10;cn=cn/10;
      daca(c==x)
            m=c*p+m;p=p*10;
   x=x+2;
pana cand !(x<10);
scrie m;
```

2. Variabila `c` memorează simultan numărul de cărți dintr-o bibliotecă (număr natural din intervalul $[3,10^2)$) și date despre fiecare carte (titlu și autor, șiruri de cel mult `20` de caractere). Expresiile C/C++ de mai jos au ca valori numărul de cărți, titlul și numele autorului celei de a treia cărți. Scrieți definiția unei structuri cu eticheta `biblio`, care permite memorarea datelor despre o bibliotecă, și declarați corespunzător variabila `c`.

`c.numar` `c.carte[2].titlu` `c.carte[2].autor` **(6p.)**

```
struct date
{
char titlu[21],autor[21];
}

struct biblio{
int numar;
struct date carte[100];
}c;
```

3. Variabila `s` poate memora un șir de cel mult `20` de caractere. Scrieți ce se afișează în urma executării secvenței alăturate. **(6p.)**

```
strcpy(s,"stilou");
cout<<s+4<<endl; | printf("%s\n",s+4);
s[0]=s[0]-1; s[1]=s[0]-3;
s[2]=s[0]+1; s[3]=s[0]+3;
s[4]='\0';
cout<<s; | printf("%s",s);
```

```
strcpy(s,"stilou");
cout<<s+4<<endl;
s[0]=s[0]-1; s[1]=s[0]-3;
s[2]=s[0]+1; s[3]=s[0]+3;
s[4]='\0';
cout<<s;
```

s="stilou"
scrie "ou"
s="rtilou" s="roilou"
s="roslou" s="rosuou"
s="rosu"
scrie "rosu"

se va afisa:

ou
rosu

SUBIECTUL al III-lea **(30 de puncte)**

1. Subprogramul **putere** are doi parametri, **n** și **p**, prin care primește câte un număr natural (**n∈[2,10⁹]**, **p∈[0,10⁹]**). Subprogramul returnează puterea la care apare numărul **p** în descompunerea în factori primi a lui **n**, dacă **p** este număr prim, sau valoarea **-1** în caz contrar.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=80** și **p=2**, subprogramul returnează numărul **4** (**80=2⁴·5**). **(10p.)**

```
int prim (unsigned long n)
{
unsigned long i;
for(i=2;i<=n/2;i++)
        if(n%i==0) return 0;
return 1;
}
```

```
int putere(unsigned long n, unsigned long p)
{
int c;
if(!prim(p)) return -1;
else
        {
        c=0;
        while(n%p==0){c++;n=n/p;}
        return c;
        }
}
```

2. O valoare **k polarizează** două șiruri dacă există doi termeni care au acea valoare, unul fiind în primul șir, iar celălalt în al doilea șir.
Scrieți un program C/C++ care citește de la tastatură numere naturale din intervalul **[1,20]**: **m**, **n** și elementele unui tablou bidimensional cu **m** linii și **n** coloane, cu proprietatea că nu există două elemente egale situate pe aceeași linie sau pe aceeași coloană. Programul afișează pe ecran valorile care pot polariza două șiruri, și anume șirul format din elementele de pe prima coloană, respectiv șirul format din elementele ultimei coloane a tabloului.
Valorile sunt afișate într-o ordine oarecare, separate prin câte un spațiu, iar dacă nu există astfel de valori se afișează pe ecran mesajul **nepolarizate**.
Exemplu: pentru **m=4**, **n=5** și tabloul alăturat se afișează pe ecran, nu neapărat în această ordine, numerele **5 6** **(10p.)**

3	7	1	2	5
2	4	5	9	6
6	2	7	8	1
5	3	2	7	8

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

int a[21][21],m,n;
void citeste()
{
        cin>>m>>n;
        int i,j;
        for(i=1;i<=m;i++)
          for(j=1;j<=n;j++)
                cin>>a[i][j];
}
```

```
void gaseste()
{
        int i,j;
        for(i=1;i<=m;i++)
        {
           for(j=1;j<=m;j++)
                if(a[i][1]==a[j][n]) cout<<a[i][1]<<' ';
        }
}

int main()
{
        citeste();
        gaseste();
        return 0;
}
```

3. Fișierul **bac.txt** conține un șir de cel mult **10⁶** numere întregi din intervalul **[-10³,10³]**, separate prin câte un spațiu. Se cere să se afișeze pe ecran suma minimă obținută adunând numere de pe poziții consecutive în șirul aflat în fișier. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare.
Exemplu: dacă fișierul **bac.txt** conține valorile **-4 6 -7 -2 1 -4 10 3 -9 -2 2**
se afișează pe ecran numărul **-12**

a. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**
b. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**

```
#include<iostream>
#include<fstream>
using namespace std;
ifstream f("bac.txt");
int s, i, smin, x, n;
int main()
{
f>>x;
smin=x;
s=x;
if(s>0)s=0;
while(f>>x)
{
        s=s+x;
        if(s<smin) smin=s;
        if(s>0) s=0;
}
cout<<smin;
return 0;}
```