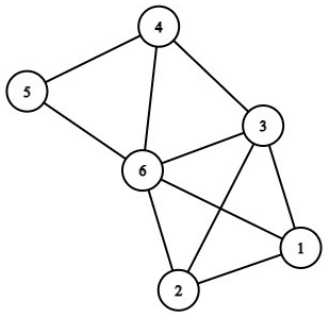
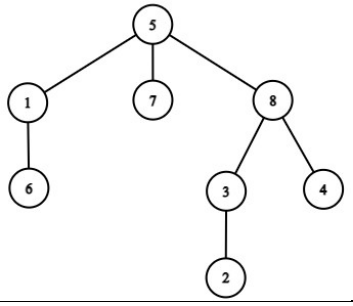
VARIANTA 5 - *Rezolvare*

Subiectul 1

•	Ultima cifră a numărului **n** este egală cu restul împărțirii sale la **10**, iar prima cifră este dată de câtul împărțirii sale la **100**, deci răspunsul corect este **b)**.
•	Deoarece variabila **q** de tip **Punct3D** conține coordonatele **x** și **y** ale unui punct din spațiu în câmpul **p** de tip **Punct2D**, rezultă că aceste coordonate se pot accesa prin expresiile **q.p.x** și **q.p.y**, în timp ce coordonata **z** poate fi accesată prin expresia **q.z**. În concluzie, răspunsul corect este **c)**.
•	Folosind metoda backtracking, primele **10** numere formate din **n=5** aflate în ordine strict crescătoare și având prima cifră cel puțin egală cu **c=3** care se vor genera sunt `34567, 34568, 34569, 34578, 34579, 34589, 34678, 34679, 34689` și **34789**, deci răspunsul corect este **a)**.
•	În figura alăturată este reprezentat graful dat în enunț. Gradul minim al unui vârf este **2** (vârful **5**), deci trebuie să eliminăm un număr minim de muchii astfel încât gradele tuturor celorlalte vârfuri să devină egale cu **2**. Se observă faptul că trebuie să eliminăm muchiile `(2,3), (6,1), (3,6)` și `(6,4)`, deci răspunsul corect este **d)**. 
•	În figura alăturată este reprezentat arborele dat în enunț. Se observă ușor faptul că răspunsul corect este **a)**. 

Subiectul 2

1.	**a)**	Algoritmul calculează câte numere formate doar din cifre pare există între **a** și **b**. Pentru precizarea valorii **4** (între **201** și **208** există **4** numere formate doar din cifre pare: **202**, **204**, **206** și **208**) se acordă **6p.**, iar pentru orice altă valoare se acordă **0p**.
	b)	Deoarece **a=818**, înseamnă că trebuie determinată cea mai mare valoare posibilă pentru **b** astfel încât între **818** și **b** să existe **7** numere formate doar din cifre pare. Se observă faptul că între **818** și **828** există **5** numere cu această proprietate (**820**, **822**, **824**, **826** și **828**), între **829** și **839** nu există nici un număr cu această proprietate (toate conțin cifra impară **3**), iar numerele **840** și **842** au proprietatea cerută. În concluzie, cea mai mare valoare posibilă pentru **b** este **843** (numărul **844** este format doar din cifre pare, deci algoritmul ar afișa valoarea **8**). Pentru precizarea valorii **843** se acordă **6p**., iar pentru orice altă valoare se acordă **0p**.

	c)	Structura repetitivă pentru **i<-a,b execută** se poate înlocui cu una dintre structurile repetitive **cât timp... execută** sau **repetă... până când**, având grijă la inițializarea, testarea și incrementarea variabilei **i**: **citește a,b** **nr ← 0** **i ← a** **┌cât timp i<=b execută** **│ d ← 1** **│ aux ← i** **│┌cât timp aux≠0 execută** **│└ c ← aux%10** **│ ┌dacă c%2 = 1 atunci** **│ │ d ← 0** **│ └■** **│ aux ← [aux/10]** **└■** **┌dacă d = 1 atunci** **└ nr ← nr+1** **└■** **i ← i+1** **└■** **scrie nr** **citește a,b** **nr ← 0** **i ← a** **┌repetă** **│ d ← 1** **│ aux ← i** **│┌cât timp aux≠0 execută** **└ c ← aux%10** **┌dacă c%2 = 1 atunci** **└ d ← 0** **└■** **aux ← [aux/10]** **└■** **┌dacă d = 1 atunci** **└ nr ← nr+1** **└■** **i ← i+1** **└până când i>b** **scrie nr** Atragem atenția asupra faptului că structura repetitivă **cât timp...execută** asigură echivalența cu structura **pentru...execută** în toate situațiile, pe când structura **repetă...până când** este echivalentă doar în unele situații, printre care și cea din acest algoritm! Pentru alegerea unei structuri repetitive se acordă **2p.**, pentru inițializarea lui **i** se acordă **1p.**, pentru testarea condiției de continuare sau oprire se acordă **3p.**, pentru incrementarea lui **j** se acordă **1p.**, iar pentru scrierea integrală a restului algoritmului se mai acordă **3p.**, deci se obțin, în total, **10p.**
	d)	Se testează capacitatea de a reprezenta pe hârtie algoritmul dat folosind un limbaj de programare studiat. Pentru instrucțiunile corecte de declarare a variabilelor, de citire a datelor, de afișare a rezultatului și de decizie se acordă câte **0.5p.**, pentru cele două instrucțiuni repetitive se acordă **2p.**, iar pentru cele 7 atribuiri se acordă **1p.** Pentru structura corectă a programului se mai acordă **1p.**, deci se obțin, în total, **6p.**
2.	În urma executării instrucțiunilor din secvența dată, șirul **s** se va transforma astfel: **"examene"->"exameneame"->"eneame"**. Pentru scrierea valorii **6** (lungimea șirului **"eneame"**) se acordă **6p.**, iar pentru orice altă valoare se acordă **0p**.	
3.	În urma apelului **f(100100,1)** se va obține valoarea **4**, deoarece **f(100100,1)=1+f(10010,1)=1+1+f(1001,1)=1+1+f(100,1)=1+1+1+f(10,1)=1+1+1+1+f(1,1)=1+1+1+1+0=4.** Pentru precizarea valorii **4**, se acordă **6p**., iar pentru orice altă valoare se acordă **0p**.	

Subiectul 3

1. Algoritmul caută, mai întâi, cel mai mic divizor propriu **d** al numărului **a**. Dacă acesta nu există, atunci parametrii **b** și **c** primesc amândoi valoarea **0**, altfel parametrul **b** primește valoarea lui **d**, iar parametrul c primește valoarea **[a/d]** (cel mai mare divizor al unui număr se obține împărțind numărul respectiv la cel mai mic divizor al său).

Limbajul Pascal

```
procedure divp(a:longint;
               var b,c:longint);
var d:longint;
begin
   d:=2;
   while((d<=a div 2)and
         (a mod d <>0)) do
     d:=d+1;
   if(d>a div 2) then
     begin
       b:=0;
       c:=0;
     end
   else
     begin
       b:=d;
       c:=a div d;
     end;
end;
```

Limbajul C++

```
void divp(int a, int &b, int &c)
{
   int d;
   d = 2;
   while((d <= a/2)&&(a%d!=0))
     d++;
   if(d>a/2)
     b=c=0;
   else
   {
      b = d;
      c = a/b;
   }
}
```

Pentru respectarea structurii antetului (**procedure**/**void**) se acordă **1p.**, pentru declararea corectă a parametrului de intrare **a** încă **1p.**, pentru declararea corectă a parametrilor de ieșire **b** și **c** încă **1p.**, pentru declararea variabilelor locale încă **1p.**, iar pentru respectarea structurii subprogramului și a sintaxei limbajului se acordă încă **1p.** Corectitudinea algoritmică a prelucrării în vederea obținerii valorii cerute este notată cu **4p.** Pentru transmiterea naturală a rezultatului prin parametrii **b** și **c** se acordă **1p**, aceasta însemnând că valorile lui **b** și **c** nu se afișează sau nu se returnează explicit. Astfel, în total, se obțin **10p.**

2. Pentru a elimina elementul aflat pe diagonala principală pe o linie **i** a matricei (respectiv elementul aflat pe coloana **j=i**), se vor deplasa cu o poziție spre stânga elementele aflate pe următoarele poziții (respectiv pe coloanele **i+1,i+2,...,n**), obținându-se astfel o matrice cu **n** linii și **n-1** coloane (am considerat faptul că liniile și coloane matricei sunt numerotate de la **0**) :

```
┌pentru i ← 0,n-1 execută
│ ┌pentru j ← i,n-2 execută
│ │  a[i][j] ← a[i][j+1]
│ └■
└■
```

Pentru declararea corectă a matricei se acordă **1p.**, pentru citirea lui **n** se mai acordă **1p.**, pentru citirea elementelor matricei se mai acordă **1p.**, pentru eliminarea corectă a tuturor elementelor aflate pe diagonala principală se mai acordă **4p.**, iar pentru

		afișarea matricei conform cerinței se acordă **2p.** În plus, pentru declararea variabilelor simple, structura și corectitudinea sintactică a programului se mai acordă **1p.**, deci se obțin, în total, **10p.**
3.	**b)**	Vom utiliza un vector de frecvențe **fc** cu **10** elemente pentru a contoriza aparițiile fiecărei cifre. Vom parcurge fișierul **bac.in** caracter cu caracter, folosind variabila **c** de tip **char**, și, în cazul în care variabila **c** conține o cifră, îi vom crește frecvența sa cu **1**. Ținând cont de faptul că un număr nu poate să înceapă cu cifra **0**, vom determina cel mai mic număr care se poate forma folosind cifrele tuturor numerelor din fișierul **bac.in** astfel: • căutăm prima cifră nenulă **cn** având frecvența cel puțin egală cu **1** (sigur există această cifră deoarece numerele din fișierul **bac.in** sunt nenule!), o scriem în fișierul de ieșire **bac.out** și îi scădem frecvența cu **1**; • începând cu cifra **0**, scriem în fișierul de ieșire toate cifrele până la **9**, fiecare cifră **c** fiind scrisă de un număr de ori egal cu frecvența sa **fc[c]**. Eficiența algoritmului din punct de vedere al timpului de executare constă în faptul că parcurgem o singură dată fișierul de intrare **bac.in**, obținând astfel un algoritm liniar în raport cu numărul de cifre din fișier. Eficiența algoritmului din punct de vedere al memoriei utilizate constă în faptul că se va folosi doar un vector de frecvențe cu **10** componente, evitând astfel memorarea tuturor celor maximum **9000000** de cifre din fișier într-un vector de caractere. Pentru o descriere coerentă a metodei se acordă **1p.**, iar pentru justificarea eficienței, încă **1p.**, deci, în total **2p.**

	a)	**Limbajul Pascal**
		
		`var fin,fout:text;`
		`  c:char;`
		`  i,j:longint;`
		`  cn:byte;`
		`  fc:array[0..9]of byte;`
		`begin`
		`  for i:=0 to 9 do`
		`  fc[i]:=0;`
		
		`  assign(fin,'bac.in');`
		`  reset(fin);`
		
		`  while(not eof(fin)) do`
		`  begin`
		`  read(fin,c);`
		`  if((c>='0')and(c<='9')) then`
		`  inc(fc[ord(c)-ord('0')]);`
		`  end;`
		
		`  close(fin);`
		
		`  assign(fout,'bac.out');`
		`  rewrite(fout);`
		
		`  cn:=1;`

```
  while (fc[cn]=0) do
    cn:=cn+1;

  write(fout,cn);
  fc[cn]:=fc[cn]-1;
  for i:=0 to 9 do
    for j:=1 to fc[i] do
      write(fout,i);

  close(fout);
end.
```

Limbajul C++

```
#include <fstream>

using namespace std;

ifstream fin("bac.in");
ofstream fout("bac.out");

int main()
{
   int fc[10],i,j,cn;
   char c;

   for(i=0;i<10;i++)
     fc[i]=0;

   while(fin>>c)
     if((c>='0')&&(c<='9'))
       fc[c-'0']++;

   fin.close();

   cn=1;
   while(fc[cn]==0)
     cn++;

   fout<<cn;
   fc[cn]--;

   for(i=0;i<=9;i++)
     for(j=0;j<fc[i];j++)
       fout<<i;

   fout.close();
   return 0;
}
```

Se acordă **1p.** pentru operațiile cu fișiere (declarare, nume corect și deschidere pentru citire), încă **1p.** pentru citirea tuturor cifrelor din fișier, **1p.** pentru un algoritm corect, încă **2p.** pentru determinarea numărului cerut și **1p.** pentru corectitudinea formală (declararea variabilelor, structura programului, sintaxa instrucțiunilor etc.).
Se acordă **1p.** pentru alegerea unui algoritm eficient ca timp de executare și **1p.** pentru utilizarea eficientă a memoriei, deci, în total, **8p.**