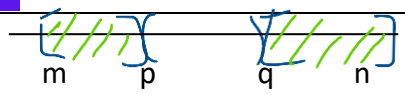
S I.1 d		x∈[m,p] U [q,n] ⇔ x>=m && x<=p || x>=q && x<=n

S I.2 C		Afișează : 2050 205 205 20 2 2 0 20 2050

F(0) n=0	Cout 0
F(2) n=2	Cout 2 If (2%10!=0) cout<<2, f(0)
F(20) n=20	Cout 20 If (20%10!=0) nu Else f(2)..... cout <<20
F(205) n=205	Cout 205 If (205%10!=0) da cout 205, f(20)
F(2050) n=2050	Cout 2050 If (2050%10!=0) NU Else f(205) .... cout<<2050

Se execută primul apel f(2050)și urmărim săgețile → apoi revenim la instrucțiunile aflate după unele apeluri urmărind săgețile - - →

S I.3 a	S P A C 1 2 3 4	Soluțiile reprezintă permutări de n elemente, la care adăugăm si condiția atribut lângă subiect (acceptăm soluțiile doar cu 13 sau 31) ~~1234, 1243~~, 1324, 1342, ~~1423, 1432~~, 2134, ~~2143~~, 2314, ~~2341~~..... Ultimele soluții sunt: ~~4123~~, 4132, 4213, 4231, 4312, ~~4321~~ 4312 este CASP (complement, atribut, subiect, predicat)
S I.4 b	M[16][16]	
S I.5 c	Un graf cu 7 noduri si 21 muchii este graf complet (un graf complet cu 7 noduri are 7*6/2 muchii=21)	Pentru a elimina nr minim de muchii, în graful parțial cu cele 2 componente conexe, trebuie ca fiecare componentă conexă să fie fiecare graf complet. Putem construi următoarele variante: 1. Prima componentă conexa cu 2 noduri, a doua componentă conexă cu 5 noduri= Vom avea nr muchii = 1+5*4/2=11 muchii→ vom elimina 10 muchii SAU 2. Prima componentă conexă cu 3 noduri, a doua cu 4 noduri= Vom avea nr muchii = 3*2/2+4*3/2=9 muchii → vom elimina 12 muchii

II.1

citește m,n	M=75, n=90							
┌pentru i←n,m,-1 execută | x←i | c←x%10 | ┌repetă | | x←[x/10] | └până când x%10≠c | ┌dacă x=0 atunci | | scrie i,' ' | └■ └■	I=90, 75,-1 I=90 X=90 C=0 Repeta X=9 Pana cand 9!=0 Da Daca x=0 NU	I=89 X=89 C=9 Repeta X=8 Pana cand 8!=9 Da Daca x=0 NU	I=88 X=88 C=8 Repeta X=8 Pana cand 8!=8 nu Repeta X=0 Pana cand 0!=8 da Daca x=0 da scrie 88	I=87 X=87 C=7 Repeta X=8 Pana cand 8!=7 Da Daca x=0 NU	I=86 X=86 C=6 Repeta X=8 Pana cand 8!=6 Da Daca x=0 NU	I=85 X=85 C=5 Repeta X=8 Pana cand 8!=5 Da Daca x=0 NU	I=84 X=84 C=4 Repeta X=8 Pana cand 8!=4 Da Daca x=0 NU	I=83 X=83 C=3 Repeta X=8 Pana cand 8!=3 Da Daca x=0 NU
	I=82 X=82 C=2 Repeta X=8 Pana cand 8!=2 Da Daca x=0 NU	I=81 X=81 C=1 Repeta X=8 Pana cand 8!=1 Da Daca x=0 NU	I=80 X=80 C=0 Repeta X=8 Pana cand 8!=0 Da Daca x=0 NU	I=79 X=79 C=9 Repeta X=7 Pana cand 7!=9 Da Daca x=0 NU	I=78 X=78 C=8 Repeta X=7 Pana cand 7!=8 Da Daca x=0 NU	I=77 X=77 C=7 Repeta X=7 Pana cand 7!=7 NU Repeta X=0 Pana cand 0!=7 da Daca x=0 da scrie 77 Daca x=0 NU	I=76 X=76 C=9 Repeta X=7 Pana cand 7!=6 Da Daca x=0 NU	I=75 X=75 C=5 Repeta X=7 Pana cand 7!=5 Da Daca x=0 NU

II.1.b) Un număr x se afișează dacă are toate cifrele egale, n∈[2222,3332]
II.1.c)

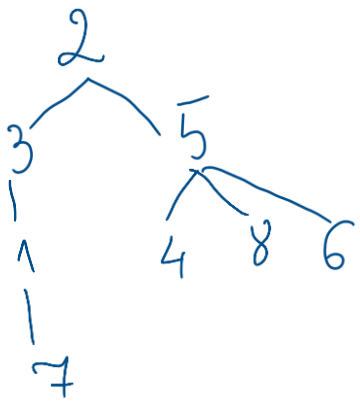
```
#include <iostream>
using namespace std;

int main()
{
   int m,n,c,x,i;
   cin>>m>>n;
   for (i=n;i>=m;i--)
   {
      x=i;
      c=x%10;
      do
      {
         x=x/10;
      }while (x%10==c);
      if (x==0)cout<<i<<" ";
   }
   return 0;
}
```

II.1.d

```
citește m,n
i←n
┌cat timp i>=m execută
| x←i
| c←x%10
| ┌repetă
| | x←[x/10]
| └până când x%10≠c
| ┌dacă x=0 atunci
| | scrie i,' '
| └■
|  i←i-1;
└■
```

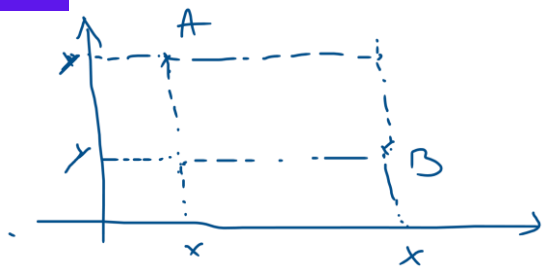
II.2

T[i] 3 0 2 5 2 5 1 5
i 1 2 3 4 5 6 7 8



Pot fi alese nodurile: 3, 1, sau 7

II.3




```
if (d.B.x-d.A.x==d.A.y-d.B.y) cout<<"DA";
else cout<<"NU";
```

III.1

```
long diviz(long n)
{
   long x,d,p;
   x=n;    d=2;
   while (n!=1)// vom descompune numărul în factori primi
      {
         p=0;//puterea factorului curent d este 0
         while (n%d==0) //împărțim la d cât timp este posibil
         {
            p++; //puterea factorului curent crește
            n=n/d;
         }
         if (p%2!=0) x=x/d;
          //de fiecare dată când găsim o putere impara, eliminăm un factor d din numărul calculat x
         d++;
      }
   return x;
}
```

III.2

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

int main()
{// vom construi șirul nou (inițial vid) astfel: adăugăm separat primul cuvânt apoi în mod repetat adăugăm " – " și un nou cuvânt extras din șirul dat.

   char s[101], nou[101]="",*p;
   cin.getline(s,101); //citim textul de la tastatură
   p=strtok(s," "); //extragem primul cuvânt p
   strcat(nou,p); //lipim primul cuvânt p la șirul construit
   p=strtok(NULL, " "); //extragem următorul cuvânt p
   while (p) //cât timp există cuvânt
   {
      strcat(nou," - ");//lipim la șirul construit spațiu, liniuță și spațiu
      strcat(nou,p);// lipim la șirul construit noul cuvânt extras, p
```

```
      p=strtok(NULL, " "); //extragem următorul cuvânt p

   }
   strcpy(s,nou);//modificăm șirul s în memorie, astfel încât acesta să conțină șirul nou construit
   cout<<s;//afișăm s
   return 0;
}
```

III.3

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("bac.txt");
long fr[101],x,y,z,maxi=0;
int main()
{
//vom construi un vector de frecventă pentru fiecare număr care poate reprezenta ipotenuza unui triunghi dreptunghic
//ipotenuza cu cele mai multe apariții va fi numărul cerut

   while (f>>x>>y>>z) //cât timp putem citi un triplet de numere x y z
   {//verificăm daca numărul x, y sau z poate reprezenta ipotenuza unui triunghi dreptunghic, iar în caz afirmativ vom crește frecvența aparițiilor ipotenuzei
      if (x*x==y*y+z*z) fr[x]++;
      else
         if (y*y==x*x+z*z) fr[y]++;
         else
            if (z*z==y*y+x*x) fr[z]++;
   }
//calculăm maximul frecvenței ipotenuzelor găsite în fișier, din vectorul de frecvență
   for (x=1;x<=100;x++)
      if (fr[x]>maxi) maxi=fr[x];
   cout<<maxi;
   return 0;
}
```

Algoritmul este eficient din punct de vedere al timpului de executare deoarece este liniar. Citim tripletele din fișier x,y,z si dacă ele pot forma un triunghi dreptunghic atunci construim frecvența ipotenuzei:

```
if (x*x==y*y+z*z) fr[x]++;
      else
         if (y*y==x*x+z*z) fr[y]++;
         else
            if (z*z==y*y+x*x) fr[z]++;
```

Frecvența maximă a ipotenuzei va reprezenta numărul cerut, care va fi afișat apoi pe ecran.