I. 1. a

2. $f(2020,0)$ 2d.

$f(202,1)$ 00

$f(20,2)$ 2100

$f(2,3)$ 022100

23022100

3. $\{1,2,3,4,5,6\}$

primele 3 poziții el. pare 3c.

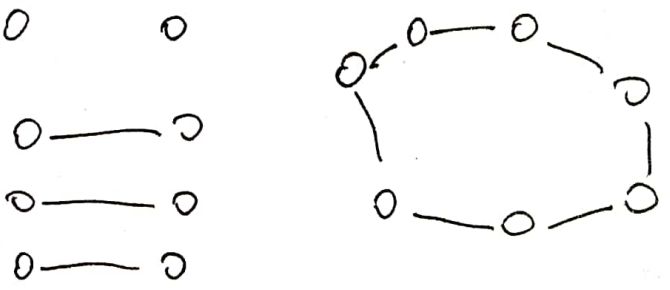
ultimele 3 poz. el. impare

a 6-a: $(2,4,6,5,3,1)$

a 7-a: $(2,6,4,1,3,5)$

4. d.

5. Graful poate fi construit astfel:



Numărul total de noduri (maxim) este 2+6+7=15.

5d.

II. a.

p	x
1	15
	5
	1
1	
	27
	9
	3
3	
	10
	3
9	

p	x
9	1
	0
	17
	5
	1
9	

//

a. 9

b. $n=2$ $\quad 4=2\cdot 2$

Un set de numere din $[0, 10^3]$ care pot fi citite pentru fi:

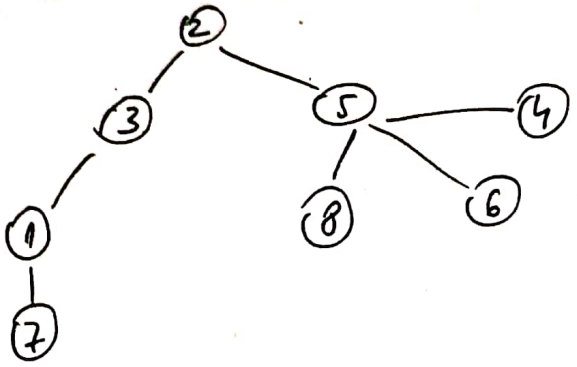
6, 18.

Două set de forme $2\cdot 3^m$, $2\cdot 3^p$; m, p naturale nenule. pentru $m \neq p$ fi ~~citite~~ citite

c.
```
#include <iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    int p=1;
    int i;
    for( i=1; i<=n; i++){
        int x;
        cin>>x;
        do{
            x = x/3;
        }while(x>3);
        if(x!=0){
            p=p*x;
        }
    }
    cout<<p;
    return 0;
}
```

d. citește n
p ← 1;
i ← 1;
cât timp $i \leq n$ execută
  citește x
  repetă
    x ← [x/3]
  până când $x \leq 3$
  dacă $x \neq 0$ atunci
    p ← p * x
  scrie p.

2.



2, 3, 1, 7

3. obs: celule care au mereu culoarea 4 au proprietatea:

$i+j < 4$ sau

$i+j > 12$.

```
for(i=0; i<9; i++)
   for(j=0; j<9; j++)
      if(i+j<4 || i+j>12) a[i][j]=4;
      else a[i][j]=2;
```

III. 1.
```
int majlung(int a, int b){
    for(int i=a; i<=b; i++){
        int si=0, sp=0;
        for(int j=2; j*j<=i; j++){
            if(i%j==0){
                int d1=j;
                int d2=i/j;
                if(d1!=d2){
                    if(d2!=1 && d2!=i){
                        if(d2%2) si+=d2;
                        else sp+=d2;
                    }
                    if(d1%2) si+=d1;
                    else sp+=d1;
                }
                else{
                    if(d1%2) si+=d1;
                    else sp+=d1;
                }
            }
        }
        if(si>sp) return i;
    }
    return 0;
}
```

2.

expl: Memorăm numărul de litere al primului cuvânt într-o variabilă n. Apoi comparăm lungimea celorlalte cuvinte cu n.

```
#include <iostream>
#include <cstring>
using namespace std;
char s[100];
int main(){
   cin.get(s,100);
   int n;
   int ok=1;
   char *p = strtok(s," ");
   n= strlen(p);
   p = strtok(NULL," ");
   while(p && ok){
      if(strlen(p)!=n){
         ok=0;
      }
      p=strtok(NULL," ");
   }
   if(ok) cout<<"DA"<<' '<<n;
   else cout<<"NU";
   return 0;
}
```

3. a.

Algoritmul memorează în variabile următoare:

- l = lungimea secvenței ascoră curente
- lmx = lungimea secvenței ascoră maximă
- ans = primul element al unei secvențe de lungime maximă
- x = valoarea curentă
- x1 = valoarea precedentă

~~Algoritmul [illegible] 0 secvențe posibile~~

Algoritmul proiectat parcurge o singură dată șirul, iar ~~dacă~~ la întâlnirea a două valori x și x1 cu proprietatea că x=x1+1, se actualizează lmx, l, ans = x1-l+1 ș.a.

Algoritmul este eficient după că fiecare element al șirului dat este folosit doar o singură dată (complexitate O(n)).

Algoritmul este eficient deoarece memoria utilizată lucrează folosește doar 5 variabile simple de tip întreg. (complexitate $O(1)$).

5.
```
#include <fstream>
#include <iostream>
using namespace std;
ifstream f("bac.txt");
int l, lmax=-1;
int ans;
int main(){
  int x;
  int x1;
  f>>x1;
  l=1;
  while (f>>x){
    if( x==x1+1) l++;
    else{
      if(lmax<l){
        ans = x1-l+1;
        lmax = l;
      }
    l=1;
    }
    x1=x;
  }
  if(lmax<l){
    ans = x1-l+1; lmax=l;
  }
  if(lmax<2) cout<<"nu exista";
  else
    for(int i=ans; i<=ans+lmax-1; i++) cout<<i<<' ';
  f.close();
  return 0;
}
```