1.

a. In programul urmator, supradefiniti operatorul << astfel incat cout<<i sa afiseze valoarea atributului i.x

b. Precizati si explicati rezultatele afisate la executarea programului astfel obtinut.

```
#include <iostream.h>
class C{
public:
  C(int i=0){x=i;}
  C& operator++(){++x; return *this;}
  C  operator--(){--x; return *this;}
private:
int x;
};
```

```
void main(){
  C i;
  cout<<i<<endl;
  cout<<++(++i)<<endl<<i<<endl;
  cout<<--(--i)<<endl<<i<<endl;
}
```

2. Adăugați o clasă pachetului alcătuit din clasa Interf și interfața ModAfisare astfel încât prin executarea programului să fie afișat mesajul

```
public class Interf{
public  static void main(String[] args){
ModAfisare ma= new Impl("Tudor");
ma.afisare("Hello");
}
}
```

```
interface ModAfisare{
public void afisare(String s);
}
```

3.

- a. Inlocuiti //1. . . , //2. . . etc. in clasa `Stack`, astfel incat metodele `push` si `pop` sa asigure tratarea exceptiilor `java.lang.Exception.`

. Numarul maxim de elemente din vectorul support este dat de expresia `suport.length.`

```
class Stack{
int varf;
     Object suport[];
     void push(Object x)//1. . .{//2. . .}
     Object pop()//3. . .{//4. . .}
     void init(int s){
          varf=0;
          support=new Object[s];
     }
     Stack(int s){//5. . .}//construieste o stiva cu s elemente
}
```

4. Explicați rezultatele afișate prin executarea fiecăreia dintre instrucțiunile //1,…//4 ale următorului program C#.

```
class A {
public virtual void M() { Console.Write("A"); }
}
class B: A {
public override void M() { Console.Write("B"); }
}
class C: B {
new public virtual void M() { Console.Write("C"); }
}
class D: C {
public override void M() { Console.Write("D"); }
}
class App{
static void Main() {
D d = new D(); C c = d; B b = c; A a = b;
d.M(); //1
c.M(); //2
b.M(); //3
a.M(); //4
}
}
```