Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1.  #include<iostream>
2.  using namespace std;
3.  class A
4.  {   int x;
5.  public:
6.      A(int i=0){x=i;}
7.      A operator+(const A& a){return x+a.x;}
8.      template <class T> ostream& operator<<(ostream&);
9.  };
10. template <class T>
11. ostream& A::operator<<(ostream& o){o<<x; return o;}
12. int main(){
13.     A a1(1), a2(2);
14.     (a1+a2)<<cout;
15.     return 0;}
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate şi modificarea)

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1  #include <iostream>
2  using namespace std;
3  class cls{
4      int v;
5  public:
6      cls(int v = 0) {this->v=v;}
7      const int& getV() const {return v;}
8  };
9  int main(){
10     cls *X = new cls(2021);
11     cout<<X->getV();
12     return 0;
13 }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate şi modificarea)

Precizați ce concept OOP/C++ se folosește în cazul următor şi dați minim 3 atribute importante ale acelui concept: utilizare, particularități, sintaxă:

Se dorește definirea unei variabile de instanță comună pentru toate obiectele din clasa C astfel încât acea variabilă să aibă aceeași valoare pentru toate obiectele din clasă, iar dacă se modifică de către un obiect să se vadă modificarea şi de către celelalte obiecte.

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1.  #include<iostream>
2.  #include<stack>
3.  using namespace std;
4.  class Test { int i;
5.  public:
6.     Test(int x = 0):i(x){cout<<"C ";};
7.     Test(const Test& x){i = x.i; cout<<"CC ";}
8.     ~Test(){cout<<"D ";} };
9.  int main() {
10.    stack<Test, vector<Test> > v;
11.    v.push_back(1);
12.    Test ob(3);
13.    v.push_back(ob);
14.    Test& ob2 = ob;
15.    v.push_back(ob2);
16. return 0;}
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se punctează doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate şi modificarea)

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1   #include<iostream>
2   using namespace std;
3
4   class B {
5       virtual void method() {
6           cout<< "B::method" <<endl;
7       }
8   public:
9       virtual ~B() {
10          method();
11      }
12      void baseMethod() {
13          method();
14      }
15  };
16  class D: B {
17      void method() {
18          cout<< "D::method" <<endl;
19      }
20  public:
21      ~D() {
22          method();
23      }
24  };
25  int main(void)
26  {
27      B* base = new D;
28      base->baseMethod();
29      delete base;
30      return 0;
31  }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate și modificarea)

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1  #include <iostream>
2  using namespace std;
3  class A{
4  public:
5      A(){cout<<"a";}
6      ~A(){cout<<"A";}
7  };
8  class B{
9  public:
10     B(){cout<<"b";}
11     ~B(){cout<<"B";}
12 }p;
13 class C:public B{
14 public:
15     C():B(){cout<<"c";}
16     ~C(){cout<<"C";}
17 }o;
18 int main(){
19     C o;
20     A b;B n;
21 }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate şi modificarea)

Precizați ce concept OOP/C++ se folosește în cazul următor și dați minim 3 atribute importante ale acelui concept: utilizare, particularități, sintaxă:

Dorim să avem un câmp constant în clasa C, în felul următor:

```
class C{
const int x;
...}
```

cum se poate inițializa acel câmp (menținând codul de mai sus neschimbat).

Se considera clasa D derivata din clasa B si se doreste instantierea unui obiect o2 din clasa D, cu datele unui obiect o1, existent din D (ex: D o2(o1)).

Precizați ce concept OOP/C++ se folosește în cazul descris anterior și dați minim 3 atribute importante ale acelui concept: utilizare, particularități, sintaxă.

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1   #include<iostream>
2   using namespace std;
3
4   class C
5   {
6       int x;
7   public:
8       C(int x = 0) : x(x) {}
9       operator float ()
10      {
11          return x;
12      }
13  };
14
15  int main ()
16  {
17      C *arr = new C[2];
18      for (int i = 1; i<2; i++) cout<<arr[i]<<" ";
19      return 0;
20  }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate şi modificarea)

Avand o metoda template f avand 2 parametri de tip T, care este diferenta intre supraincarcarea cu o functie non-template cu parametrii de tip int si o supraincarcarea cu o functie template cu tipul explicit int? In cazul unui apel a lui f cu ambii parametrii de tip int care din functii se apeleaza? Dar in cazul unui apel cu un parametru de tip int si unul double?

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1   #include<iostream>
2   using namespace std;
3
4   class A {
5       int i;
6   public:
7       A(int x=-8):i(x) {}
8       virtual int f(A a) {
9           return i+a.i;
10      }
11  };
12  class B: public A {
13      int j;
14  public:
15      B(int x=2):j(x) {}
16      int f(B b) {
17          return j+b.j;
18      }
19  };
20  int main()
21  {
22      int i=20;
23      A *o;
24      if (i%4) {
25          A a;
26          o=new A(i);
27      }
28      else {
29          B b;
30          o=new B(i);
31      }
32      cout<<o->f(*o);
33      return 0;
34  }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate și modificarea)

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1   #include<iostream>
2   using namespace std;
3   class C
4   {
5   public:
6       C() {}
7       ~C ()
8       {
9           cout<< "~C";
10      }
11  };
12  int main()
13  {
14      try
15      {
16          throw "exception";
17          C c;
18      }
19      catch(const char* s)
20      {
21          cout<<s;
22      }
23      return 0;
24  }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate și modificarea)

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1   #include<iostream>
2   using namespace std;
3
4   class A
5   {
6       static int x;
7       const int y=10;
8   public:
9       A(int i)
10      {
11          x=i;
12          x=-y+3;
13      }
14      int put_x(A a)
15      {
16          return x+a.y;
17      }
18  };
19  int A::x=8;
20  int main()
21  {
22      A a(2);
23      cout<<a.put_x(26);
24      return 0;
25  }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate și modificarea)

Avand o functie cu un parametru, cum pot transmite ca parametru actual, un obiect din clasele D1 si D2 derivate dintr-o aceeasi clasa B (astfel ca daca in functie, parametrul formal apeleaza o metoda f din clasa de baza B, sa se execute metoda f din clasa corespunzatoare a parametrului actual).

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1   #include<iostream>
2   using namespace std;
3
4   class A {
5   protected:
6       int nm;
7   public:
8       A(int hbr = 1) : nm(hbr) { std::cout << "??"; }
9       int ha() { return nm; }
10      virtual void r() const {};
11      virtual ~A() {};
12  };
13
14  class B : public A {
15      int d;
16  public:
17      B(int b = 2) : d(b) { std::cout << "!"; }
18      void r(int z) const  { std::cout << nm << " " << z << "\n"; }
19  };
20
21  void warranty(const A& am) {
22      am.r();
23  }
24
25  int main() {
26      A ha;
27      B un(ha.ha());
28      warranty(un);
29  }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se puncteaza doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate și modificarea)

Pentru urmatorul program scris in C++ cu flagurile de compilare default din codeblocs (g++ bazat pe GNU gcc)

```
1  #include <iostream>
2  using namespace std;
3  class A{
4  protected:
5      int n;
6  public:
7      A(){n=12;}
8      int getn(){return n;}
9      void addn(int x){n+=x;}
10     virtual ~A(){};
11 };
12 class B:public A{
13 protected:
14     int m;
15 public:
16     B():A(){m=12;}
17     int getm(){return m;}
18 };
19 int main(){
20     A *x=new A;
21     x->addn(x->getn());
22     B *y=dynamic_cast<B*>(x);
23     cout<<y->getn();
24 }
```

Spuneti daca programul compilează? DA / NU

Dacă DA ce se afișează pe ecran (Atentie: se punctează doar daca se explica in detaliu cum se ajunge la rezultatul / rezultatele afisate)

Dacă NU: de ce nu?

Modificarea care îl face să meargă (o singură linie modificată, de precizat numărul liniei modificate și modificarea)

Pot intoarce ca rezultat prin referinta, o data nestatica a unui obiect ce este parametru al unei metode? Dar o data statica? Discutie dupa modul de transmitere al parametrului.