1 Specificați si testați funcția: (1.5p)

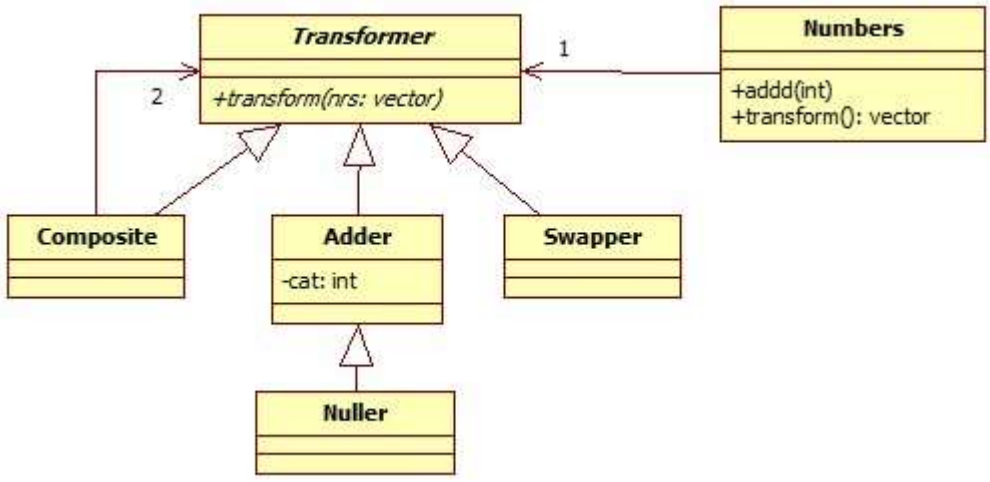
```
bool f(int a) {
     if (a <= 0)
          throw std::exception("Illegal argument");
     int d = 2;
     while (d<a && a % d>0) d++;
     return d>=a;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Dacă sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
using namespace std;
class A {
public:
 A(){cout << "A()" << endl;}
 void print() {cout << "printA" <<
endl;}
};
class B: public A {
public:
 B(){cout << "B()" << endl;}
 void print() {cout << "printB" <<
endl;}
};
int main() {
      A* o1 = new A();
      B* o2 = new B();
      o1->print();
      o2->print();
      delete o1;delete o2;
   return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
#include <vector>
using namespace std;
int main() {
      vector<int> v;
      v.push_back(5);
      v.push_back(7);
      v[0] = 6;
      v.push_back(8);
      auto it = v.begin();
      it = it + 1;
      while (it != v.end()) {
            cout << *it << endl;
            it++;
      }
   return 0;
}
```

3 Scrieți codul C++ ce corespunde diagramei de clase UML. **(4p)**



- Clasa abstracta **Transformer** are o metoda pur virtuala transform(nrs)
- Metoda transform() din clasa **Adder** adaugă la fiecare număr un număr dat (cat) , metoda transform din **Swapper** interschimbă numere consecutive (poziția 0 cu poziția 1, poziția 2 cu 3, etc) iar transform() din clasa **Nuller** înlocuiește numărul cu 0 daca in urma aplicări adunării numărul este > 10 sau lasă numărul ce rezulta in urma adunării. Clasa **Composite** in metoda transform() aplica succesiv cele doua transformări folosind **Transfromer**-ele agregate.
- Metoda transform() din clasa **Numbers** ordonează descrescător numerele adăugate cu add si apelează metoda transform(nrs) din Transformer-ul continut.

Se cere:

1 Codul C++ **doar pentru clasele**: **Transformer**,**Composite**, **Nuller (0.75p)**
2 Scrieți o funcție fiecare creează si returnează un obiect **Numbers** care compune un Nuller (cat=9) cu un Swapper compus cu un Adder (cat=3). **(0.5p)**
3 In funcția main apelați funcțiile de mai sus, adăugați cate 5 numere in cele doua obiecte **Numbers**. apoi apelați funcția transform pentru ambele. **(0.25p)**

- Creați doar metode si atribute care rezulta din diagrama UML (adăugați doar lucruri specifice C++ ex: constructori). Nu adăugați câmpuri, metode, nu schimbați vizibilitatea, nu folosiți friend. **Barem**: **1.5p** Polimorfism, **1p** Gestiunea memoriei, **1.5p** Defalcat mai sus

4 Definiți clasele ToDo si Examen general astfel încât următoarea secvență C++ sa fie corecta sintactic si sa efectueze ceea ce indică comentariile. (2p)

```
void todolist() {
	ToDo<Examen> todo;
	Examen oop{ "oop scris","8:00" };
	todo << oop << Examen{"oop practic", "11:00"};	//Adauga 2 examene la todo
	std::cout << oop.getDescriere();//tipareste la consola: oop scris ora 8:00
	//itereaza elementele adaugate si tipareste la consola lista de activitati
	//in acest caz tipareste: De facut:oop scris ora 8:00;oop practic ora 11:00
	todo.printToDoList(std::cout);
}
```