1 Specificați si testați funcția: **(1.5p)**

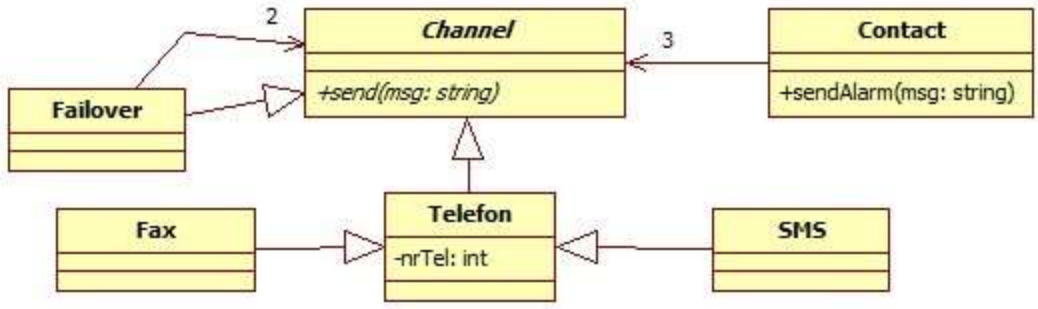
```
bool f(int a) {
	if (a <= 1)
		throw "Illegal argument";
	int aux = 0;
	for (int i = 2; i < a; i++) {
		if (a % i == 0) {
			aux++;
		}
	}
	return aux == 0;
}
```

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

```
//2 a  (1p)
#include <vector>
#include <iostream>
using namespace std;
class A {
public:
	virtual void f() = 0;
};
class B:public A{
public:
	void f() override {
		cout << "f din B";
	}
};
class C :public B {
public:
	void f() override {
		cout << "f din C";
	}
};
int main() {
	vector<A> v;
	B b;
	v.push_back(b);
	C c;
	v.push_back(c);
	for (auto e : v) { e.f(); }
	return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
public:
 A() {cout << "A" << endl;}
 ~A() {cout << "~A" << endl;    }
 void print() {cout << "print" <<
endl;}
};
void f() {
	A a[2];
	a[0].print();
}
int main() {
	f();
	return 0;
}
```

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



- Clasa abstractă **Channel** are o metoda pur virtuala *send*
- Metoda *send* din clasa **Telefon** tipărește mesajul "dail:" si numărul de telefon conținut, dar din când in când (in funcție de un număr aleator generat) aruncă excepție std::exception indicând ca linia este ocupată.
- Clasa **Fax** si **SMS** încearcă sa apeleze numărul de telefon si in caz de succes tipărește "sending fax" respectiv "sending sms". Clasa **Failover** încearcă sa trimită mesajul pe primul canal, dacă trimiterea eșuează (este ocupat) atunci încearcă trimiterea pe canalul secundar.
- Metoda *sendAlarm* din clasa **Contact**, încearcă sa trimită repetat mesajul pe cele 3 canale conținute pe rând până reușește trimiterea (găsește o linie care nu este ocupat).

Se cere:

1 Codul C++ **doar pentru clasele**: **Channel**, **Failover**, **Fax, Contact(0.75p)**

2 Scrieți o funcție C++ care creează si returnează un obiect **Contact** cu următoarele canale (alegeți voi numere de telefon): 1 Telefon; 2 Fax "daca este ocupat încearcă" Sms ; 3 Telefon "daca este ocupat încearcă" Fax "daca este ocupat încearcă" SMS. **(0.5p)**

3 In funcția main apelați funcția de mai sus si trimiteți 3 mesaje. **(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. Folosiți STL unde exista posibilitatea.

Detalii barem: **1.5p** Polimorfism, **1p** Gestiunea memoriei, **1.5p** Restul(Defalcat mai sus)

4 Definiți clasa Expresie generală astfel încât următoarea secvență C++ sa fie corecta sintactic si să efectueze ceea ce indică comentariile. **(2p)**

```
void operatii() {
	Expresie<int> exp{ 3 };//construim o expresie,pornim cu operandul 3
	//se extinde expresia in dreapta cu operator (+ sau -)  si operand
	exp = exp + 7 + 3;
	exp = exp - 8;
	//tipareste valoarea expresiei (in acest caz:5 rezultat din 3+7+3-8)
	cout << exp.valoare()<<"\n";
	exp.undo();  //reface ultima operatie efectuata
	//tipareste valoarea expresiei (in acest caz:13 rezultat din 3+7+3)
	cout << exp.valoare() << "\n";
	exp.undo().undo();
	cout << exp.valoare() << "\n";  //tipareste valoarea expresiei (in acest caz:3)
}
```