1 Specificați si testați funcția: (1.5p)

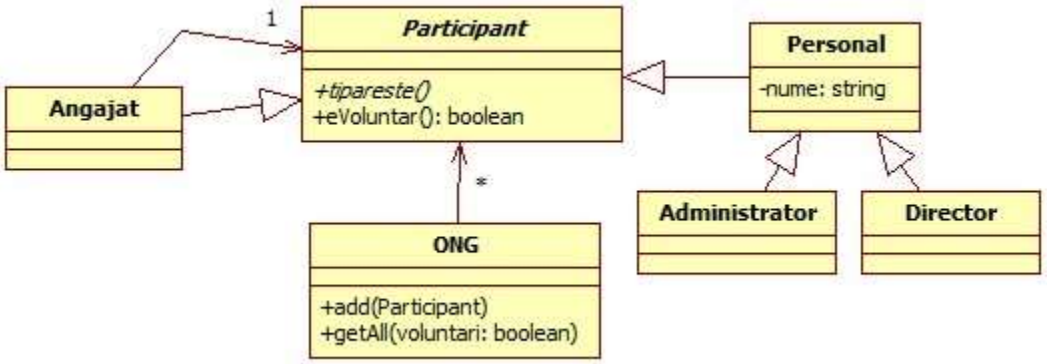
```
using namespace std;
#include <vector>
#include <string>
#include <algorithm>
#include <map>
vector<int> f(vector<int> l) {
	if (l.size() == 0)
		throw exception("Illegal argument");
	map<int, int> c;
	for (auto e : l)  {
		c[e]++;
	}
	sort(l.begin(), l.end(), [&](int a, int b) {
		return c[a] > c[b]; });
	return l;
}
```

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>
class A {
public:
	A() {
		std::cout << "A";
	}
	virtual void print() {
		std::cout << "printA";
	}
};
class B : public A {
public:
	B() {
		std::cout << "B";
	}
	virtual void print() {
		std::cout << "printB";
	}
};
int main() {
	std::vector<A> v;
	A a;
	B b;
	v.push_back(a);
	v.push_back(b);
	for (auto e : v) {e.print();}
	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[1].print();
}
int main() {
	f();
	return 0;
}
```

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



- Clasa abstracta **Participant** are o metoda pur virtuala *tipareste*.
- Metoda *tipareste* din clasa **Personal** tipărește numele persoanei.
- Clasa **Administrator** si **Director** pe lângă ce tipărește clasa de baza mai tipărește si cuvântul “Administrator” respectiv “Director” .
- Clasa **Angajat** tipărește, pe lângă ce tipărește personalul agregat de el, si textul “ angajat”. Metoda eVoluntar returnează false.
- Metoda *add* din clasa **ONG** permite adăugarea de orice participant, iar metoda *getAll* returnează doar participanții angajați sau participanții voluntari (in funcție de parametru). Implicit toți participanții sunt voluntari daca nu sunt decorate cu **Angajat**.

Se cere:

1 Codul C++ **doar pentru clasele**: **Participant**, **Angajat**, **Director, ONG(0.75p)**

2 O funcție C++ care creează si returnează un obiect **ONG** si adaugă următorii participanți (alegeți voi numele pentru fiecare): un administrator voluntar, un administrator angajat, un director voluntar si un director angajat. **(0.5p)**

3 In funcția main tipăriți separat angajații si voluntarii din ONG. **(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 Cos generală astfel încât următoarea secvența C++ sa fie corecta sintactic si sa efectueze ceea ce indica comentariile. **(2p)**

```
void cumparaturi() {
      Cos<string> cos;//creaza un cos de cumparaturi
      cos = cos + "Mere"; //adauga Mere in cos
      cos.undo();//elimina Mere din cos
      cos + "Mere"; //adauga Mere in cos
      cos = cos + "Paine" + "Lapte";//adauga Paine si Lapte in cos
      cos.undo().undo();//elimina ultimele doua produse adaugate

      cos.tipareste(cout);//tipareste elementele din cos (Mere)
}
```