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

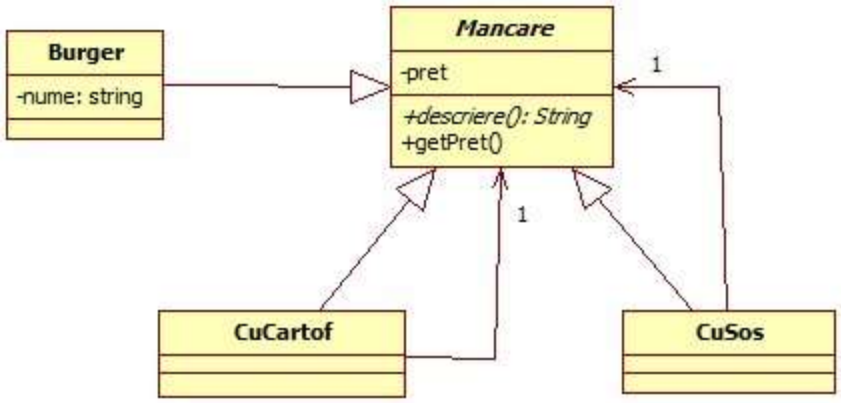
```
void f(vector<int>& l, int poz) {
	if (poz < 0 || poz >= l.size()) throw exception{};
	int a = 0;
	int b = l.size()-1;
	int nr = l[poz];
	while (a < b) {
		while (a<b && l[a] < nr ) a++;
		while (b>a && l[b] > nr) b--;
		if (a < b) {
			swap(l[a], l[b]);
			if (l[a] == l[b] && l[b] == nr) a++;
		}
	}
}
```

**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:
	virtual void print() = 0;
};
class B : public A {
public:
	virtual void print() {
		std::cout << "printB";
	}
};
class C : public B {
public:
	virtual void print() {
		std::cout << "printC";
	}
};
int main() {
	std::vector<A> v;
	B b; C c;
	v.push_back(b);
	v.push_back(c);
	for (auto e : v) { e.print(); }
	return 0;
}
```

```
//2 b  (0.5p)
void f(bool b) {
	std::cout << "1";
	if (b) {
		throw
std::exception("Error");
	}
	std::cout << "3";
}
int main() {
	try {
		f(false);
		f(true);
		f(false);
	}
	catch (std::exception& ex) {
		std::cout << "4";
	}
	return 0;
}
```

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




- Clasa abstracta **Mancare** are o metoda pur virtuala descriere()
- **CuCartof** si **CuSos** conțin o mâncare si metoda descriere() adaugă textul "cu cartof" respectiv "cu sos" la descrierea mâncării conținute. Prețul creste cu 3 RON pentru cartofi, mâncarea cu sos costa in plus 2 RON. Clasa **Burger** reprezintă un hamburger fără cartof si fără sos, metoda descriere() returnează denumirea hamburgerului.

**Se cere:**

1 Codul C++ **doar pentru clasele**: **Mancare**, **Burger**, **CuSos (0.75)**
2 Scrieți o funcție C++ care returnează o lista de mâncăruri: un burger „McPuisor", un burger „BigTasty" cu cartof si sos, un burger „Booster" cu cartof si un burger „Booster" cu sos (alegeți voi prețul de baza pentru fiecare mâncare). **(0.5)**
3 In programul principal creați o lista de mâncăruri (folosind funcția descrisa mai sus), apoi tripartiți descrierea si prețul pentru fiecare in ordinea descrescătoare a preturilor. **(0.25)**

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 si static. Folosiți STL unde exista posibilitatea. Detalii barem: **1.5p** Polimorfism, **1p** Gestiunea memoriei**, 1.5p** Restul (defalcat mai sus)

**4** Definiți clasa Conferinta si Sesiune astfel încât următoarea secvență C++ să fie corectă sintactic si să efectueze ceea ce indică comentariile. **(2p)**

```
int main() {
	Conferinta conf;
	//add 2 attendants to "Artifiial Inteligente" track
	conf.track("Artifiial Inteligente") + "Ion Ion" + "Vasile Aron";
	//add 2 attendants to "Software" track
	Sesiune s = conf.track("Software");
	s + "Anar Lior" + "Aurora Bran";
	//print all attendants from group "Artifiial Inteligente" track
	for (auto name : conf.track("Artifiial Inteligente")) {
		std::cout << name << ",";
	}
	//find and print all names from Software track that contains "ar"
	vector<string> v = conf.track("Software").select("ar");
	for (auto name : v) { std::cout << name << ","; }
}
```