Test C++

Nota: Cateva dintre intrebari care se refera la clase, constructori, proprietati, metode etc cer si cate un exemplu. Puteti sa faceti doar 1-2 clase la care sa faceti referire cand vi se cere sa scrieti un astfel de exemplu (adica sa nu construiti cate o alta clasa pentru fiecare exemplu cerut, ca sa nu pierdeti timp). De asemenea, pentru exercitiile care cer sa gasiti gresala, nu se considera gresala faptul ca lipseste un include pentru o biblioteca.

1. a)Se considera urmatorul cod; ce va afisa?
```
#define A 5
#define B A+A
std::cout<<"B"<<B-B;
```

   b)Daca avem:
```
#define F(X) X-1
```
   Scrieti ce vor afisa urmatoarele linii de cod:
```
std::cout<<2*F(4);
std::cout<<2*F(F(4));
```

2. Ce va afisa programul? Explicati.
```
char sir[] = "abcdefg";
int lg = strlen(sir);
for (int i = 0; i < lg;i++)
      std::cout << sir[*(sir)+i-'a']-*(sir+i);
```

3. Ar putea fi corecta urmatoarea linie de cod?
   a=f(a=5,3)?22:1,2,3;
   Daca veti considera ca o astfel de instructiune nu poate fi niciodata corecta, explicati de ce.
   Daca veti considera ca e corecta, explicati ce face si dati un exemplu.
4. Se da urmatorul cod:
```
#include <iostream>
using namespace std;
class Punct
{
      public:
            int x;
            int y;
      Punct(int _x, int _y):x(_x),y(_y){}

      //definiti operatorul delete
      //definiti operatorul +
      //definiti operatorul ==

      void afis(){
            cout<<"punct("<<x<<","<<y<<")"<<endl;
      }
};

int main(int argc, char* argv[])
{
```

```
	Punct a(10,10), b(20,20), c(15,15);
	(a+b).afis();
	cout<<endl<<((a+b)==c);
	return 0;
}
```

Definiti(supraincarcati) operatorul delete astfel incat sa afiseze textul “Am sters dinamic un punct” la fiecare apel. Definiti operatorul + astfel incat sa calculeze punctul de mijloc al segementului format de cele doua puncte (media aritmetica a coordonatelor). Atentie coordonatele sunt intregi (ne putem imagina niste puncte care se refera la pixeli sau alte unitati nedivizibile). Definiti operatorul == care verifica faptul ca doua puncte au coordonate egale.

5. Explicati ce sunt unique_ptr, shared_ptr, weak_ptr, auto_ptr si dati exemple de folosire.
6. Scrieti un program scurt in care il definiti pe B, inlocuind *[...]* cu ce veti considera necesar:
   `#define B […]`
   Programul trebuie sa contina si linia de cod:
   `cout<< (B==B+1) <<"\n";`
   Scrieti programul de asa natura incat aceasta linie de cod sa afiseze 1 pe ecran. Puteti sa va definiti functii si variabile dupa cum aveti nevoie.
7. Care din urmatoarele variabile se poate folosi pentru a memora un vector de siruri? Se considera rezolvata problema numai daca sunt enumerate toate variantele corecte si numai acelea. Atentie, unele variante pot contine erori de sintaxa, ceea ce le face evident gresite.

```
char* a[10];
char** b;
char*** c;
char d[10][0];
char* e[20][10];
char* &f;
char g[][10];
char h[10][10];

typedef char* sir;
sir k[3];
sir *p;
```

8. Ce este un destructor? Poate un destructor sa fie static? Explicati de ce.

Scrieti un destructor pentru clasa C_b din exemplul de mai jos astfel incat sa nu avem memory leaks:

```
class C_a
{
	int x;
	public:
		C_a(int _x):x(_x){}
};

class C_b
{
	C_a** y;
	int n;
	public:
		C_b(int _n):n(_n)
		{
			y=new C_a*[n];
			for(int i=0;i<n;i++)
				y[i]=new C_a(i);

		}
};
```

9. Consideram fisierele header: *a.h*, *b.h*, *c.h*. Atat *b.h*, cat si *c.h* il includ pe *a.h*:
   `#include “a.h”`
   …
   Fisierul fis.cpp are nevoie si de functiile definite in *a.h*, dar si de cele din *c.h*, asadar le va

include pe amandoua:

```
#include “b.h”
#include “c.h”
```

Ce problema apare? Cum putem rezolva?

10. Se considera secventa de cod:

```
int b=1;
int x=(b++)+(b++);
```

Ce vor afisa instructiunile urmatoare?

```
std::cout<<x;
std::cout<<b;
```

11. Se considera clasa:

```
class C
{
	static int x;
	int y;
	//.... eventual alte proprietati si metode
	public:
		static void f()
		{
			x=y;
		}
		void g() const
		{
			y=x;
		}
};
```

Ce este gresit in definirea functiei f? Dar in functia g?

12. Ce va afisa programul? Explicati calculul.

```
#include <iostream>
using namespace std;
#define HEHE CONCAT
#define A ret
#define B urn
#define CONCAT(a,b) a ## b
#define AB(a,b) HEHE(a,b)
#define C 2
#define D 2
int f()
{
	int ret=1;
	int urn=5;
	A=urn+C;
	B=D+ret;
	A=B;AB(A,B)+(A+B+C+D);
	#define AB 5
	A=A+B+AB;
	return A;
}

int main(int argc, char* argv[])
{
	cout<<f();
	return 0;
}
```

13. A)Se considera functia:

```
int f(int a, int b) {…}
```

care din urmatoarele functii reprezinta un overload pentru functia f? Explicati raspunsurile.

```
a) void f(int a) {...}
```

```
b) class C{
   float f(char *b) {...}
}
c) float f(int a, int b) {...}
```

B) Avem supraincarcarile pentru functia sum:

```
void afis(int a) {
      std::cout<<"int: "<<a;
}

void afis(float a) {
      std::cout<<"float: "<<a;
}

void afis(double a) {
      std::cout<<"double: "<<a;
}
```

Vrem sa apelam functia afis pentru valoarea 3. Scrieti cate un apel pentru fiecare functie afis, cu aceasta valoare si explicati cum ati facut ca sa va asigurati ca intra in functia dorita si nu in alta varianta a ei. Ce se intampla daca apelam afis('a') ?

14. Se considera urmatorul cod:

```
int f(float a, double *b, int t)
{
      return int(a + *b * t);
}
//definirea lui g...
int main()
{
      float x=3.7;
      g=f;
      int y=g(&x,2.5, 2);
      std::cout<<y;
      return 0;
}
```

Cum ar trebui sa arate definirea variabilei g ca sa fie corect codul? Ce tip de date trebuie sa aiba?

15. Consideram programul:

```
class C1 {
      static int x;
public:
      virtual void afis1() {
            std::cout << "parinte"<<std::endl;
      }

      void afis2() {
            std::cout << "parinte2" << std::endl;
      }
};

class C2:public C1{

public:
      void afis1() override {
            std::cout << "deriv"<<std::endl;
      }
      void afis2() {
            std::cout << "deriv2" << std::endl;
```

```
        }
};

void f_afis1(C1 *elem) {
        elem->afis1();
}
void f_afis2(C1 *elem) {
        elem->afis2();
}

int main() {
        C2 b;
        b.afis1();
        ((C1)b).afis1();
        f_afis1(&b);
        b.afis2();
        ((C1)b).afis2();
        f_afis2(&b);
}
```

Explicati cum de merge conversia (C1)b. Explicati ce afiseaza programul.

16. Ce este un namespace? Dati un exemplu. Avand secventa de cod de mai jos:

```
namespace N1{
        namespace N2{
                class C1{
                        class C2{
                                static void f(){
                                        std::cout<<"ceva"<<std::endl;
                                }
                        };
                };
         };
};
```

Putem apela functia f? Daca da, dati un exemplu de apel. Daca nu, corectati codul, fara a schimba structura namespace-urilor si a claselor si apoi dati un exemplu de apel.

17. Consideram apelul de functie:

```
f(&v,&x);
std::cout<<v;
```

Declarati v si x si definiti functia f, astfel incat functia f sa creeze un vector (de caractere) cu literele mici de la a la z, pe care sa il puna in v. Iar in x sa puna numarul de caractere generate (cate litere sunt de la a la z). Cand construiti vectorul de caractere aveti in vedere si faptul ca afisarea de dupa apelul lui f trebuie sa puna pe ecran sirul abc...z. (deci cu ce trebuie sa se termine v?)

18. Consideram programul:

```
int t;
std::cin >> t;
switch (t) {
        case 1:std::cout << "unu";
        case 2:std::cout << "doi";
        case 3:case 4:std::cout << "trei sau patru";
        default: std::cout << "altceva";
}
```

Vrem ca daca citim 1 sa afiseze “unu”, daca citim 2, sa afiseze “doi”, daca citim 3 sau 4, sa afiseze “trei sau patru” si daca citim alt numar, sa afiseze “altceva”. Face acest lucru? Daca nu, corectati.

19. Se considera programul:

```
#include <iostream>
struct Ceva{
	int v[3];
	int a;
	void afiseaza_struct()
	{
		for(int i=0;i<3;i++)
			std::cout<<"v["<<i<<"]="<<v[i]<<std::endl;
		std::cout<<"a="<<a<<std::endl;
	}
};

int main()
{
	Ceva c;
	for (int i=0;i<4;i++)
		c.v[i]=(i+1)*100;
	c.afiseaza_struct();
	return 0;
}
```

Ce va afisa acest program la rulare? Explicati raspunsul.

20. Dorim sa avem doua clase Persoana si Birou. Pentru o persoana vrem sa avem un camp cu un pointer catre obiectul Birou de care aceasta tine. De asemenea clasa Birou contine un camp de tip Persoana* ca fiecare instanta sa aiba un pointer catre seful biroului. Codul ar arata asa:

```
class Persoana {
	Birou* b;
};

class Birou {
	Persoana* sef;
};
```

Ce problema observati? Cum o putem rezolva?

21. Care este diferenta dintre structura si clasa? Dati un scurt exemplu din fiecare.
22. [STL] Consideram un vector:

```
std::vector<int> v;
```

Presupunem ca am inserat niste elemente in el. Scrieti o functie f care primeste ca argument un vector de acest fel si afiseaza dimensiunea totala pe care ocupa datele sale in memorie.

23. [STL] Definiti un map cu chei caractere si valori int. Inserati cateva elemente in map(vreo 3-4). Parcurgeti map-ul folosind un iterator si afisati perechile cheie valoare in formatul: key → value. De exemplu:

```
'a'->3
```

PROBLEMA PRACTICA

(Obiective: lucrul cu fisiere, parsare, structuri de date, operatori)
Inainte de a incepe rezolvarea, cititi tot enuntul.
Scrieti un program in C++ care indeplineste urmatoarele cerinte.
Se citesc dintr-un fisier de intrare numele unor directoare sau fisiere. Informatiile pot sa apara fie sub linii de forma:

```
Director -> element
```

Unde element poate fi un alt director sau fisier, precizat prin director(nume), respectiv fisier(nume). Aceasta linie indica faptul ca element il are ca folder parinte pe Director.
Fie sub linii de forma:

```
fisier(20)
```

Unde fisier e numele fisierului iar 20 e dimensiunea sa in kb (pentru a nu ne complica, vom presupune ca toate fisierele au ca dimensiune un numar intreg de kb, de aceea nu mai precizam si unitatea de masura in datele de intrare).
Consideram de asemenea ca in numele directoarelor sau fisierelor nu avem caracterele: (, >,).
Directoarele radacina(care nu au parinte) vor fi memorate intr-un vector.
Veti crea o clasa Fisier cu campurile id (nr. natural), nume(sir de caractere), dimensiune (int)). Se va crea si o clasa numita Director care extinde clasa **Fisier.**
Aceasta va avea acces la proprietatile nume si id, insa nu si la dimensiune.
Clasa Director va avea un camp in plus Fisier** cu ajutorul caruia va memora un array de pointeri catre obiectele Fisier care sunt continute de director.
In clasa Fisier se va crea o functie virtuala e_director() care returneaza false. Aceasta functie va fi redefinita (suprascrisa) in clasa Director, astfel incat sa returneze true.

Clasa Fisier va avea o metoda de afisare (care va scrie valorile tuturor proprietatilor). Clasa Director va suprascrie si aceasta functie, afisand in plus si numele fisierelor continute de directorul respectiv.

Mentiuni despre clasa Fisier:

- Niciuna dintre proprietatile clasei Fisier nu va fi publica.
- Memorarea numelor se va face in mod eficient, folosind vectori de caractere alocati dinamic.
- Veti defini pentru id, nume si dimensiune atat un setter cat si un getter. In functiile setter se va verifica faptul ca numele e un sir de litere si cifre (nu si caractere nealfanumerice -vom impune aceasta conditie asupra numelor fisierelor). In cazul in care setterul primeste o valoare incorecta, “arunca” o eroare.
- Clasa Fisier va avea un constructor fara parametri, un constructor care primeste doar numele ca parametru, si unul care primeste si numele si dimensiunea . Pentru nume face validarea enuntata mai sus. Id-ul va fi dat in mod automat fiecarui obiect, astfel: pentru primul obiect id-ul va fi 1, pentru al doilea, 2 etc.
- Veti scrie un destructor care elibereaza memoria (valabil si pentru clasa Director).
- erorile aruncate de setteri si constructori vor fi tratate cu blocuri de tip try-catch in program.

  Definiti operatorul + pentru director. Acesta va primi un operand de tip Fisier si va adauga in lista sa de fisiere, in caz ca nu exista deja, adresa obiectului Fisier respectiv. Acest operator va fi folosit pentru a adauga fisierele citite, din datele de intrare. Se va definisi operatorul - pentru director, care primeste ca operand un id (numar intreg) si sterge Fisierul cu acel id din lista de fisiere-fii. Veti face si o metoda de calculare a dimensiunii unui Director. Dimensiunea totala a

acestuia consta in suma dimensiunilor fisierelor continute de el si de directorarele continute de el (in mod recursiv, e vorba si de subdirectoarele directoarelor). Aceasta metoda va arunca o eroare in caz ca nu se poate calcula acest lucru(unul din fisiere nu are dimensiunea cunoscuta). Un director vid are dimensiune 0.
Dupa ce au fost citite toate datele din fisier se va exemplifica folosirea operatorului – pentru un id oarecare si pentru directoarele radacina (cele care nu au parinte) se vor afisa datele si dimensiunea totala.

Exemplu fisier de intrare:
d1 -> director(d2)
d3 -> director(d1)
d1 -> fisier(f1)
f1(20)
d4 -> fisier(f2)
d3 -> fisier(f3)
f3(10)
d2 -> fisier(f4)
f4(5)