TEST C++
NUME:
GRUPA:

1. **Fie dat urmatorul program:**

```
#include <iostream.h>
using namespace std;
class c1    { public: int a; } ;
class c2 : private c1
{
public:
        c1::a;
        int b;
        void scrie_a(){ cout << "a = " << a << endl;}
} ;
void main()
{       c2 ob; ob.scrie_a (); }
```

Selectati afirmatia corecta:

a. functia de acces nu are drept de acces la membrul ***a*** deoarece derivarea s-a realizat private;
b. programul afiseaza valoarea lui ***a***;
c. functia are doar acces *read-only* asupra unui membru dintr-o clasa derivata privat;
d. derivarea privata este incorect realizata;
e. prin derivare privata, accesul la membrii mosteniti ramane privat.

2. **Se dă clasa :**

```
class c  {
        double a, b ;
public :
        friend c operator + (c &, double ) ;
        friend c operator + (double, c & ) ;
};
```

In declaratia de mai sus apare:

a. supraincarcare prin functii independente;
b. supraincarcare prin functii membre in clasa c;
c. supraincarcare de constructori de clasa c;
d. supraincarcare de constructori de copiere ai clasei c;
e. declararea unor clase *friend*.

3. **Se dă structura :**

```
#include <iostream.h>
using namespace std;
struct persoana
{
        char nume[50];
        struct copil { char prenume[10]; int virsta;  } c[3];
} p1,*pp;

void main()
{
        pp=&p1; p1.c[0].virsta=5;
        cout << pp->c->virsta;                          // 1
        cout << pp->c[0].virsta;                // 2
        cout << (pp->c+1)->virsta;                      // 3
        cout << ((*pp).c+1)->virsta;                    // 4
        cout << (*pp).c[0].virsta;              // 5
        cout << (*pp).c->virsta;                // 6
}
```

Care din expresiile:

pp->c->virsta; // 1

pp->c[0].virsta; // 2
(pp->c+1)->virsta; // 3
((*pp).c+1)->virsta; // 4
(*pp).c[0].virsta; // 5
(*pp).c->virsta; // 6

afiseaza valoarea 5?

a. toate
b. 2+3+5
c. 2+3+5+6
d. 1+2+5+6
e. 1+2+4+5+6

4. Fie declaratia :

```
class c1   { /* ... */ } ;
class c2 : public c1    {/*... */} ;
```

Atunci clasa c2 fata de c1 este:

a) derivata; b) de baza; c) friend; d) virtuala; e) derivata friend

5. Fiind data clasa:

```
#include <conio.h>
#include <iostream>
using namespace std;
class persoana {

public:
          float salariu;
          persoana(float s=0): salariu(s){ }
          operator float( ) { return salariu; }
          float indexare(float coef)
                              { return salariu *(1+coef/100); }
};

void main( )
{   persoana p(100); cout << p.indexare(p); getch(); }
```

Apelul functiei indexare( ):

a. foloseste cast-ul definit de programator;
b. genereaza eroare, prin folosirea recursiva a obiectului p;
c. genereaza eroare, nexistînd o supraîncarcare ce primeste obiect persoana;
d. se traduce prin indexare(int);
e. se traduce prin indexare(void);

6. Având declaratia :

```
class persoana {
          int virsta ;
public :
          persoana () ;
          int spune_virsta() { return virsta ; }
};
```

Functia int spune_virsta( ) :

a. asigura acces read only la un membru privat al clasei;
b. este o functie friend;
c. este o metoda obisnuita;
d. asigura acces read - write la un membru privat al clasei;
e. este eronata, deoarece variabila virsta este privata.

7. Având o clasa definita astfel:

```
class ex {
int a ;
public :
        friend ostream& operator << (ostream& , ex  ) ;
};
```

Functia friend ostream& operator << (ostream& , ex) supraîncarca operatorul << ?

a. nu, neavînd suficienti parametri.
b. da, în scopul citirii datelor obiectului;
c. da, pentru implementarea operatiei de deplasare la stinga, pe obiecte;
d. nu, operatorul << neputindu-se supraincarca;
e. da, în scopul afisarii datelor obiectului;

8. Fie clasa :

```
class c {
        int a, b ;
public :
        c (int , int ) ;
        int det_a ( ) {return a ;}
        ~c () ;
};
```

Semnul ~ are rolul :

a. de a nega pe biti rezultatul returnat de metoda c( );
b. de a preciza existenta destructorului;
c. de a nega logic rezultatul returnat de metoda c( );
d. de a supraincarca constructorul clasei;
e. de a supraincarca operatorul ~

9. Se dau urmatoarele clase:

```
#include <iostream>
using namespace std;

class B1 { int x; };
class B2 { int y; };
class B3 { int z; };
class B4 { int t; };
class  D: public B1, private B2, protected B3, B4  { public: int m; };

void main()
{
        D d;
        cout << d.m; // varianta 1
        cout << d.x; // varianta 2
        cout << d.y; // varianta 3
        cout << d.z; // varianta 4
        cout << d.t; // varianta 5
}
```

Variantele care au acces la variabile pentru afisare sunt:

a) 1 + 2 + 4 + 5 b) 1 + 2 c) 1 + 2 + 4 d) 1 e) 1 + 2 + 5

10. Se dă clasa :

```
#include <iostream>
using namespace std;
#include <string.h>

class person
{
        double wage;
```

```
    public:
        char name[20];
        person(char n[]=" anonymous",double w=0):wage(w)
        {               strcpy(name, n); }
        person * gA()
        {               return this;      }
    };

void main()
{
        person p1=person("Daniel",5000), p2("John",3500);
        cout << endl  << (&p1)->name;                    // varianta  1
        cout << endl  << &p1->name;                      // varianta  2
        cout << endl << p2.gA()->name <<"\n\n";          // varianta  3
        cout << endl << p2.this->name <<"\n\n";   // varianta  4
}
```

Care din variantele de mai jos afiseaza corect numele unei persoane ?

a) 2+3 b) 1+2 c) 1+3 d) 1+2+3 e) 1+2+3+4

11. Fiind data urmatoarea secventa de cod:

```
#include <iostream>
using namespace std;

class persoana{
private:
        float salariu;
public:
        char nume[20];
        virtual float calc_sal( )  {return 0.;}          //  Salariu persoanei
};
class inginer : public persoana{
public:
        float calc_sal( )    {return 1.;}                // Salariu în regie
};
class muncitor : public persoana{
public:
        float calc_sal( )   {  return 2.;}               // Salariu în acord
};

void main( )
{
        persoana p, *pp; inginer i,*pi; muncitor m, *pm;
    pp = &p; pi=&i; pm=&m;

    pp=pi; cout << endl <<pp->calc_sal( );
    pp=pm; cout << endl <<pp->calc_sal( );

    p=i;  cout << endl << p.calc_sal( );
    p=m; cout << endl << p.calc_sal( );
}
```

programul afiseaza in ordine, valorile:

a) 0 0 1 2 b) 1 2 1 2 c) 0 1 0 2 d) 0 0 0 0 e) 1 2 0 0

12. Se da programul:

```
#include <iostream>
using namespace std;
class c{
        int a;
```

```
public :
        c() {}
        c(const c&);
        c& operator =(c&);
};

c& c::operator=(c &c){ cout << endl  << "copiere cu egal"; return c;}
c::c(const c &c) { cout << endl << "Constructor de copiere"; }
void main()
{
        c x,y=x;
        c b=x;  x=y;
};
```

programul:

a. apeleaza de doua ori operator=(), o data constructorul de copiere si o data constructorul implicit;
b. apeleaza de trei ori constructorul de copiere, o data constructorul implicit;
c. apeleaza de trei ori supraincarcarea operatorului =;
d. apeleaza de doua ori constructorul de copiere si de trei ori operator=();
e. apeleaza de doua ori constructorul de copiere, o data operator=() si o data constructorul implicit;

13. Pentru urmatorul program:

```
#include <iostream>
using namespace std;
class persoana{
public:
        int virsta;
        persoana(int v=30) : virsta(v){}
};
class profesor{
public:
        int virsta;
        profesor(int v=20) : virsta(v){}
    operator persoana(){ persoana p; p.virsta = virsta; return p; }
};

persoana f(persoana p)  { p.virsta++; return p; }

void main()
{
   persoana p; p=f(p);   cout << endl << p.virsta;
   profesor prof; p=f(prof);          cout << endl << p.virsta;
}
```

varstele afisate la rularea programului de mai sus sunt:

a. 30 20, ambele obiecte fiind temporare, la transmiterea prin valoare;
b. 31 21, datorita incrementarilor din functie;
c. 31 20, profesor fiind temporar, datorita conversiei implicite prin cast;
d. 30 21, persoana fiind temporar, datorita conversiei implicite prin cast;
e. 0 0, deoarece pentru obiecte temporare s-au apelat constructori de copiere.

14. Programul următor:

```
#include <iostream>
using namespace std;
class Clasa1{
     bool value;
public:
     virtual void f(){cout<<"Clasa1"<<endl;}
```

```
};
class Clasa2{
	int caracter;
	int articol;
public:
	void f(){cout<<"Clasa2"<<endl;}
};
void egale(){cout<<"sizeof(obC1) = sizeof(obC2)"<<endl;}
void obC1obC2(){cout<<"sizeof(obC1) < sizeof(obC2)"<<endl;}
void obC2obC1(){cout<<"sizeof(obC2) < sizeof(obC1)"<<endl;}
void main()
{
	Clasa1 obC1;
	Clasa2 obC2;
	void(*f)();
	sizeof(obC1) == sizeof(obC2)? f = egale : sizeof(obC1) < sizeof(obC2) ? f = obC1obC2 : f =
obC2obC1;
	(*f)();
}
```

tipãreste:

a. sizeof(obC2) < sizeof(obC1);
b. sizeof(obC1) < sizeof(obC2);
c. sizeof(obC1) = sizeof(obC2);
d. generează eroare la compilare datoritã unei indirectãri incorecte;
e. generează eroare la compilare deoarece operatorul conditional ?: nu permite efectuarea unor atribuiri.

15. In programul următor:

```
#include <iostream>
using namespace std;
class Persoana{
	int varsta;
	char* nume;
public:
	Persoana(int v=0, char* n="Oarecare"):varsta(v){
		this->nume = new char[strlen(n)+1];
		strcpy(this->nume,n);
		cout<<"Constructor"<<endl;}
	Persoana(Persoana& p){
		this->varsta = p.varsta;
		this->nume = new char[strlen(p.nume)+1];
		strcpy(this->nume, p.nume);
		cout<<"Constructor de copiere"<<endl;}
	void operator=(Persoana& p){
		this->varsta = p.varsta;
		delete[] this->nume;
		this->nume = new char[strlen(p.nume)+1];
		strcpy(this->nume, p.nume);
		cout<<"Operator="<<endl;}
	~Persoana(){ cout<<"Destructor"<<endl;}
};

void main()
{
	Persoana p1, p2(20, "Gigel");
	Persoana p3 = p1;
	p3 = p2;
	Persoana p4 = p1;
}
```

sunt apelate următoarele:

a. constructor – de patru ori, constructor de copiere – o datã, destructor – de patru ori;
b. constructor – de trei ori, constructor de copiere - de douã ori, destructor de cinci ori;
c. constructor – de douã ori, constructor de copiere – de douã ori, operator= - o datã, destructor – de patru ori;
d. constructor – de douã ori, constructor de copiere – o datã, operator= - de douã ori, destructor – de douã ori;
e. constructor – de douã ori, constructor de copiere – o datã, operator= - de douã ori, destructor – de patru ori.

16. De câte ori este apelat destructorul clasei Persoana în programul urmãtor?

```
#include <iostream>
using namespace std;
class Persoana{
public:
    Persoana() {cout<<"Constructor"<<endl;}
    ~Persoana() {cout<<"Destructor"<<endl;}
};

void main(){
        Persoana** ppp;
        ppp = new Persoana*[5];
        for(int i=0; i<5; i++)
        ppp[i] = new Persoana();
        //prelucrari
        for(int i=0; i<5; i++)
        delete ppp[i];
}
```

Raspuns:
a. 10;
b. 6;
c. 7;
d. 5;
e. niciunul din rãspunsurile anterioare.

17. Sã se precizeze ce afiseazã programul urmãtor:

```
#include <iostream>
using namespace std;
class Masina{
        int anFabr;
        char* culoare;
public:
        Masina(int an = 0, char* cul = ""){
                this->anFabr = an;
                this->culoare = new char[strlen(cul)+1];
                strcpy(this->culoare, cul);
        }
Masina& operator=(Masina& m){
        this->anFabr = m.anFabr;
        delete[] this->culoare;
        this->culoare = new char[strlen(m.culoare)+1];
        strcpy(this->culoare, m.culoare);
        return (*this);
        }
int getAnFabr() {return this->anFabr;}
void setAnFabr(int anFabricatie) {this->anFabr = anFabricatie;}
char* getCuloare() {return this->culoare;}
void setCuloare(char* c){
        delete[] this->culoare;
        this->culoare = new char[strlen(c)+1];
        strcpy(this->culoare, c);
        }
};
```

```
void main() {
        Masina m1(2000,"Alb");
        Masina m2(2001,"Negru");
        Masina m3 = m2;
        Masina m4(2003, "Rosu");
        m3.setCuloare("Verde");
        m4 = m1;
        m4.setCuloare("Albastru");
        cout<<m1.getAnFabr()<<" "<<m1.getCuloare()<<" ; ";
        cout<<m2.getAnFabr()<<" "<<m2.getCuloare()<<" ; ";
        cout<<m3.getAnFabr()<<" "<<m3.getCuloare()<<" ; ";
        cout<<m4.getAnFabr()<<" "<<m4.getCuloare()<<" ; ";
}
```

afiseazã:

a. 2000 Albastru ; 2001 Negru ; 2001 Verde ; 2000 Albastru ;
b. 2000 Alb ; 2001 Negru ; 2001 Verde ; 2000 Albastru ;
c. 2000 Alb ; 2001 Negru ; 2001 Negru ; 2000 Albastru ;
d. 2000 Alb ; 2001 Verde ; 2001 Verde ; 2000 Albastru ;
e. Niciun rãspuns corect

18. Se consideră clasa

```
class Student{
public:
        char * nume;
        int note[10];
int nrnote;
...
Student(int *v, int dim,char* num){...}
Student operator=(Student s){
nume = new char[strlen(s.nume)+1];
strcpy(nume,s.nume);
for(int i=0;i<s.nrnote;i++) note[i] = s.note[i];
nrnote = s.nrnote;
return *this;
}
~Student(){
        if(nume) delete[]nume;}
};
```

Execuţia programului următor:

```
void main()
{
int vector[] = {1,2,3};
        Student s1(vector,3,”Popescu”);;
        Student s2(vector,3,”Gigel”);
        s1 = s2;
}
```

are ca efect:

a. Copierea valorilor din s2 în s1 cu generare de memory leak datorită câmpului *nume*.
b. Copierea valorilor din s2 în s1 fară generare de memory leak
c. Copierea valorilor din s2 în s1 cu generare de memory leak datorită câmpului *note* care nu este dezalocat în destructor.
d. Eroare la execuţia operatorului = deoarece nu este alocat spaţiu pentru câmpul *note*.
e. Eroare la apelul constructorului cu parametrii nefiind respectat tipul parametrilor de intrare.

19. Pentru a defini corect (fără a genera în aplicaţii viitoare memory leaks, pointeri cu valoarea 0xcccccc sau iniţializări de pointeri diferiţi cu aceeaşi adresă) clasa Student ce are atributele

```
char nume[30];
int *note;
int nrNote
```

este nevoie obligatoriu de:

a. constructor cu parametrii/fără parametrii, constructor copiere, operator =;
b. constructor cu parametrii, constructor de copiere, destructor
c. constructor fără şi cu parametrii, constructor copiere, operator =, destructor
d. constructor cu parametrii, constructor implicit, operator = , destructor
e. constructor copiere, operator =, destructor

20. Dacă se consideră clasa

```
class Test{
private:
        int valoare;
        Test(int vb){valoare = vb;}
public:
int GetValoare() { return valoare;}
}
```

si forma supraîncărcată a operatorului +

```
int operator +(int vb, Test t){
return vb+t.GetValoare();
}
```

analizaţi instrucţiunile:

```
Test t(5);
int vb = 10 + t
```

a. Sunt corecte si vb ia valoare 15
b. Sunt corecte si vb ia valoarea 10;
c. Nu sunt corecte deoarece operatorul + nu este anuntat ca fiind *friend* in clasa
d. Nu sunt corecte deoarece operatorul trebuie supraîncărcat numai prin funcţie membră
e. Nu sunt corecte deoarece atributul *valoare* este privat si nu este accesibil din afara clasei

Grila	Raspuns
1	B
2	A
3	D
4	A
5	A
6	A
7	E
8	B
9	D
10	C
11	E
12	E
13	B
14	B,C
15	C
16	D
17	D
18	A
19	C
20	A