Programare Orientată Obiect, Tip-C, Sem-1, Zi (2020-2021)

Început la	Wednesday, 2 December 2020, 16:12
Status	Terminat
Completat la	Wednesday, 2 December 2020, 16:13
Timp luat	31 secs
Notează	**0,00** din maxim 5,00 (**0%**) posibil

NAVIGARE ÎN TEST

1 2 3 4 5

Afișați câte o pagină pe rând

Finalizare verificare

1 Întrebare

Nu a primit răspuns

Marcat din 1,00

Întrebare cu flag

Being given the next source code (the 3 called functions are defined correctly) indicate what it will print, **if function1() executes throw MyException();**

```
int main() {
    int vb = 10;

    try {
        vb = 20;
        function1();
        vb = 30;
        function2();
        vb = 40;
        function3();
        vb = 50;
    }
    catch (MyException ex) {
        vb = 60;
    }
    catch (MyException* pEx) {
    }
    catch (int ex) {
        vb = ex;
    }
    catch (...) {
        vb = 80;
    }
    cout << endl << vb;
}
```

Alegeți o opțiune:

a. 10

b. 20

c. 30

d. 40

e. 50

f. 60

g. 70

h. 80

Your answer is incorrect.

2 Întrebare

Nu a primit răspuns

Marcat din 1,00

Întrebare cu flag

Being given the next source code (the 3 called functions are defined correctly) indicate what it will print, **if function1() executes throw "wrong value"**

```
int main() {
    int vb = 10;

    try {
        vb = 20;
        function1();
        vb = 30;
        function2();
        vb = 40;
        function3();
        vb = 50;
    }
    catch (MyException ex) {
        vb = 60;
    }
    catch (MyException* pEx) {
    }
    catch (int ex) {
        vb = ex;
    }
    catch (...) {
        vb = 80;
    }
    cout << endl << vb;
}
```

Alegeți o opțiune:

a. 10

b. 20

c. 30

d. 40

e. 50

f. 60

g. 70

h. 80

Your answer is incorrect.

3 Întrebare

Nu a primit răspuns

Marcat din 1,00

Întrebare cu flag

Being given the next source code (the 3 called functions are defined correctly) and the MyException class indicate what it will print, **if function2() will execute throw new MyException();**

```
18  int main() {
19      int vb = 10;
20
21      try {
22          vb = 20;
23          function1();
24          vb = 30;
25          function2();
26          vb = 40;
27          function3();
28          vb = 50;
29      }
30      catch (MyException ex) {
31          vb = 60;
32      }
33      catch (MyException* pEx) {
34      }
35      catch (int ex) {
36          vb = ex;
37      }
38      catch (...) {
39          vb = 80;
40      }
41      cout << endl << vb;
42  }
```

Alegeți o opțiune:

a. 10

b. 20

c. 30

d. 40

e. 50

f. 60

g. 70

h. 80

Your answer is incorrect.

4 Întrebare

Nu a primit răspuns

Marcat din 1,00

Întrebare cu flag

Being given the next source code (the 3 called functions are defined correctly) indicate what it will print, **if function 3 will execute throw 70;**

```
18  int main() {
19      int vb = 10;
20
21      try {
22          vb = 20;
23          function1();
24          vb = 30;
25          function2();
26          vb = 40;
27          function3();
28          vb = 50;
29      }
30      catch (MyException ex) {
31          vb = 60;
32      }
33      catch (MyException* pEx) {
34      }
35      catch (int ex) {
36          vb = ex;
37      }
38      catch (...) {
39          vb = 80;
40      }
```

```
41      cout << endl << vb;
42  }
```

Alegeți o opțiune:

a. 10

b. 20

c. 30

d. 40

e. 50

f. 60

g. 70

h. 80

i. you can't throw a number; it will generate compiler error

Your answer is incorrect.

5 Întrebare

Nu a primit răspuns

Marcat din 1,00

Întrebare cu flag

Being given the next source code (the 3 called functions are defined correctly) indicate what it will print, **if neither of the 3 functions is NOT throwing anything** (they just do some computations)

```
18  int main() {
19      int vb = 10;
20
21      try {
22          vb = 20;
23          function1();
24          vb = 30;
25          function2();
26          vb = 40;
27          function3();
28          vb = 50;
29      }
30      catch (MyException ex) {
31          vb = 60;
32      }
33      catch (MyException* pEx) {
34      }
35      catch (int ex) {
36          vb = ex;
37      }
38      catch (...) {
39          vb = 80;
40      }
41      cout << endl << vb;
42  }
```

Alegeți o opțiune:

a. 10

b. 20

c. 30

d. 40

e. 50

f. 60

g. 70

h. 80

Your answer is incorrect.

Finalizare verificare

Programare Orientată Obiect, Tip-C, Sem-1, Zi (2020-2021)

Început la	Wednesday, 18 November 2020, 15:44
Status	Nu a fost trimis niciodată

1 întrebare
Răspuns salvat
Marcat din 1,00
Întrebare cu flag

When is the copy constructor called (choose one or more answers) ?

Alegeţi una sau mai multe opţiuni:
- a. When you pass an object by pointer to a function call
- b. When you receive an object by reference from a function call (the function returns the object reference)
- c. When you create a new object based on existing one using ClassType obj2 = obj1 pattern
- d. When you receive an object by value from a function call (the function returns the object)
- e. When you pass an object by value to a function call
- f. When you pass an object by reference to a function call

2 întrebare
Răspuns salvat
Marcat din 1,00
Întrebare cu flag

A class member function can always access the data in ________ , (in C++).

Alegeţi o opţiune:
- a. the object of which it is a member
- b. the public part of its class
- c. the class of which it is member
- d. the private part of its class

3 întrebare
Răspuns salvat
Marcat din 1,00
Întrebare cu flag

If you have a x64 application (for a x64 processor) and the next definition double* values = new double[10], what is the size in bytes of values ?

Alegeţi o opţiune:
- a. 80
- b. 8
- c. 4
- d. 10
- e. 40

4 întrebare
Răspuns salvat
Marcat din 1,00
Întrebare cu flag

If we have (check the given class Card) we can use ________ to access int s:

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

class Card {
public:
   int s;
};

int main() {
 Card a;
 Card* p = &a;
}
```

Alegeţi o opţiune:
- a. p->int
- b. p->s
- c. p.s
- d. a.int
- e. a->s

5 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Being given the next class and main() implementation, please say how many times the destructor is called before reaching line 16 ?

```
1   #include <iostream>
2   using namespace std;
3
4   class Test {
5       int attr1;
6       char attr2;
7   };
8
9   int main() {
10      Test tObject;
11      Test values[100];
12      Test* newValues = new Test[10];
13
14      delete[] newValues;
15
16      cout << endl << "Phase 1 end";
17  }
```

Alegeţi o opţiune:
- a. 0
- b. 111
- c. 10
- d. We don't have a default destructor
- e. 100

6 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

What is the console output of the following program?

```
#include <iostream>
using namespace std;
main() {
   char s[] = "hello", t[] = "hello";
   if(s==t)
      cout<<"equal strings";
}
```

Alegeţi o opţiune:
- a. equal strings
- b. unequal strings
- c. no output
- d. Compilation error

7 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

What defines the public interface of a class definition ?

Alegeţi o opţiune:
- a. collection of public methods
- b. only the collection of public attributes
- c. only the collection of public static methods
- d. the class description
- e. collection of public and private attributes

8 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Being given the next class Test, what will be the output/result of compiling and running the sequence ?

```
class Test {
    int x;
};
int main()
{
  Test t;
  cout << t.x;
  return 0;
}
```

Alegeţi o opţiune:
- a. 0
- b. Garbage/Residual value
- c. Runtime Error
- d. Compiler Error

9 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Being given the next class and main() implementation, please say how many times the default constructor is called before reaching line 16 ?

```
1   #include <iostream>
2   using namespace std;
3
4   class Test {
5       int attr1;
6       char attr2;
7   };
8
9   int main() {
10      Test tObject;
11      Test values[100];
12      Test* newValues = new Test[10];
13
14      delete[] newValues;
15
16      cout << endl << "Phase 1 end";
17  }
```

Alegeţi o opţiune:
- a. We don't have a default constructor
- b. 10
- c. 111
- d. 1
- e. 100

10 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Which of the following(one or more answers) is/are NOT part of an object internal structure?

Alegeţi o opţiune:
- a. Private static variable
- b. Private variable (any type and not static)
- c. Public static variable
- d. Static variable
- e. Static constant variable

11 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

What is the output of the following program?

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

main() {
   int const a = 5;

   a++;
   cout<<a;
}
```

Alegeţi o opţiune:
- a. 5
- b. Compile error
- c. 6
- d. Runtime error

12 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Being given the next class and main() implementation, please say how many times the copy constructor is called before reaching line 16 ?

```
1   #include <iostream>
2   using namespace std;
3
4   class Test {
5       int attr1;
6       char attr2;
7   };
8
9   int main() {
10      Test tObject;
11      Test values[100];
12      Test* newValues = new Test[10];
13
14      delete[] newValues;
15
16      cout << endl << "Phase 1 end";
17  }
```

Alegeţi o opţiune:
- a. We don't have a default copy constructor
- b. 0
- c. 111
- d. 10
- e. 100

13 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Assume a x86 processor. Also, assume that there is no alignment in objects. Predict the output following program.

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

class Test
{
    static int x;
    int *ptr;
    int y;
};

int main()
{
    Test t;
    cout << sizeof(t) << " ";
    cout << sizeof(Test *);
}
```

Alegeţi o opţiune:
- a. 12 8
- b. 8 4
- c. 8 8
- d. 12 4
- e. 12 12

14 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

For a class Student what is the "this" variable type ?

Alegeţi o opţiune:
- a. Student*
- b. a float
- c. an integer
- d. Student
- e. char*

15 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

When one object reference/pointer variable is assigned to another object reference/pointer variable then

Alegeţi o opţiune:
- a. a copy of the object is created. You have 2 different objects with the same values
- b. a copy of the reference is created. You have only one object
- c. it is illegal to assign one object reference/pointer variable to another object reference/pointer variable.
- d. a copy of the reference is not created. You have 2 different objects with the same values referenced by 2 different pointers

16 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Which of the following CANNOT be passed to a function in C++ ?

Alegeţi o opţiune:
- a. Structure
- b. Array
- c. Header file
- d. Pointer to a variable
- e. Constant

17 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

What is a "dangling pointer" in C++ ?

Alegeţi o opţiune:
- a. A pointer initialized with 0XCCCC....CC
- b. A normal pointer
- c. A pointer initialized with nullptr
- d. A pointer initialized with NULL
- e. A pointer that stores an address from HEAP that does not belong anymore to your processor
- f. A pointer that stores a valid address in HEAP

18 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Predict the output of following C++ program

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

class Empty {};

int main()
{
  cout << sizeof(Empty);
  return 0;
}
```

Alegeţi o opţiune:
- a. 0
- b. Compiler error
- c. Runtime Error
- d. A non-zero value

19 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

An object is __________ of a class.

Alegeţi o opţiune:
- a. an interface
- b. a member function
- c. an instance
- d. an encapsulation

20 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

What is the difference between struct and class in C++?

Alegeţi o opţiune:
- a. All of statements are true
- b. All members of a structure are public and structures don't have copy constructors
- c. Members of a class are private by default and members of struct are public by default.
- d. All members of a structure are public and structures don't have constructors and destructors

Finalizare verificare

NAVIGARE ÎN TEST

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Afişaţi câte o pagină pe rând
Finalizare verificare

Grile POO strict secrete

1)In terms of pseudo-code with what sequence of operators is the following statement equivalent?

A: a1.operator = (operator+=(10,a2));

2)
```
#include <iostream>
using namespace std;

class Base {
public:
	int x, y;

public:
	Base(int i, int j) { x = i, y = j; }
};

class Derived : public Base {
public:
	Derived(int i, int j) {
		this->x = i;
		this->y = j;
	}

	void print() {
		cout << x << " " << y;
	}
};

 int main() {

	 Derived q(10, 10);
	 q.print();



}
```

A:Compiler error on Derived Constructor

3) **A method in a derived/subclass that has the same signature as another method in the parent/base class:**

A: will override the base class method

4)

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

class Base {
public:
	void show() {
		cout << " I am a Base";
	}
};

class Derived : public Base {
public:
	int value = 5;
	void show() {
		cout << "I am a Derived";
	}
};


int main(void)
{
	Base* bp;
	Derived d;
	bp = &d;
	bp->show();
	cout << bp->value;
}
```

A: compiler error on line 25 (`cout << bp->value;`)

6)

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

class Base1 {
public:
	Base1()
	{
		cout << " Basel ctor" << endl;
	}
	};

class Base2 {
public:
	Base2()
	{
		cout << "Base2 ctor" << endl;

	}
};

class Derived : public Base1, public Base2 {
public:
	Derived()
	{
		cout << "Derived ctor" << endl;
	}
};

		int main(){
			Derived d;
	}
```

A:

Base1 ctor

Base2 ctor

Derived ctor

7)When the inheritance is public, the private methods in base class are _____ in the derived class (in C++).

A: inaccessible

8) Catch blocks must:
A: be used immediately after the try block

9)

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


class Base {

public:
	virtual void print() {
		cout << endl << "This is Base";
	}
};

class Derived : public Base {
public:
	virtual void print() {
		cout << endl << "This is Derived";
	}
};

void printInfo(Base p) {
	p.print();
}


int main() {
	Base b;
	Derived d;
	printInfo(b);
	printInfo(d);
}
```

A:
This is Base
This is Base

10)

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

```
class ClassA {
public:
	ClassA() {
		cout << endl << "Normal Constructor";
	}
	ClassA(const ClassA& a)
	{
		cout << endl << "Copy constructor";
	}
};
int main() {

	ClassA* t1, * t2;
	t1 = new ClassA();
	t2 - new ClassA(*t1);
	ClassA t3 = *t1;
	ClassA t4;
	t4=t3;
};
```

A:
Normal Constructor
Copy constructor
Copy constructor
Normal Constructor

11) When a copy constructor may be called?

A:
When an object of the class is passed (to a function) by value as an argument
When an object of the class is returned by value from a function
When an object is constructed based on another object of the same class

12)How does C++ compiler differs between overloaded postfix and prefix operators?

A: A postfix ++ has a dummy parameter

13) What is the significance of the friend keyword in C++?

A: it is used to give access to global methods or other classes on the current class private area;

14) What is the difference between the role of the operator = and copy constructor?

A: copy constructor creates new object, the = operator works with two existing objects;

15) What is THIS inside a C++ class constructor?

A: Pointer that manages the built object address

16) Which of the following are not inherited?

A: Constructors and Destructors

17) What is DOWN- casting?

A: converting base type pointers or objects to subclass type pointers or objects

18)

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


class ClassA {
public:
	ClassA() {
		cout << endl << "Constructor called";
	}
};

int main() {

	ClassA a1;
}
```

A: Constructor called

19) Fill up the next sequence. Constructors have ____ implicit return type.

A: Its class type

20) What is the role of accessor methods in a class?

A: Provide access in read/ write mode to private class attributes;

C++ questions for self evaluation

1. What is the size of an int variable (consider a 32/64 bit VS2015 compiler)?
   4 bytes

2. What is the size of a short int variable (a 32/64 bit VS2015 compiler is considered)?
   2 bytes

3. What is the size of a long int variable (a 32/64 bit VS2015 compiler is considered)?
   4 bytes

4. What is the size of a long long int variable (a 32/64 bit VS2015 compiler is considered)?
   8 bytes

5. What is the size of a bool variable (a 32-bit VS2015 compiler is considered)?
   1 byte

6. What is the size of a float variable (a 32-bit VS2015 compiler is considered)?
   4 bytes

7. What is the size of a double variable (a 32-bit VS2015 compiler is considered)?
   8 bytes

8. What is the size of a long double variable (a 32-bit VS2015 compiler is considered)?
   8 bytes

9. What is the size of a char variable (a 32-bit VS2015 compiler is considered)?
   1 byte

10. What is the size of a pointer to int - int * (a 32-bit VS2015 compiler is considered)?
    4 bytes

11. What is the size of a pointer to char - char * (a 32-bit VS2015 compiler is considered)?
    4 bytes

12. What is the keyword *this* in C ++?
    A pointer received as a parameter by all the non-static methods of a class, that contains the address of the object that called the method.
    - The address of the object that calls a class member function
    - The address of the object created in the constructor
    - The address of the object destroyed in the destructor

13. What does represent *this* in the constructor?
    A pointer that contains the address of the object to be created

14. What does represent *this* in the destructor?
    A pointer that contains the address of the the object to be destroyed

15. What does represent *this* in a class member function?
    A pointer that contains the address of the object that calls the function

16. What does represent *this* in a static function?
    A static function is NOT associated with any objects => it does NOT receive the *this* pointer

17. What does represent *this* in the copy constructor?
    A pointer to the object to be created

18. What does represent *this* in the overloaded operator =?
    A pointer to the object to be modified

19. What is a member function?
    A function whose definition/prototype is in the definition of a class (like any other variable) and operates on any object/instance of the respective class, and has access to all the members of that class

20. What is a static function?
    A member function, independent from the objects if the class.
    A static function can be called even if there are no instances of the class. They are called using the class name and the resolution operator.
    Static functions can directly access other static members (variables and functions), but not non-static members.

    e.g.

```
class className
{
        Static int value;
Public:
        Static int getValue()
        {
                Return value;  //static member fct
        }

        int className::value{ 1 }       //initializer
```

21. What is a static attribute?
    An attribute that isn't part of the object, but shared by all objects of the class.
    Static attributes can't be initialized inside a class.

e.g.

```
class className {
public:
        static int objectCount;
}
Int className::objectCount = 0;
```

22. What is a constant attribute?
    An attribute that cannot be modified when creating the object, once it has been initialized.

23. When can a constant attribute be initialized?
    When creating the object, in the constructor initialization list.

24. What does overload mean? (supraincarcarea)
    The ability to have multiple functions with the same name, but with a different parameter list/number.

25. What is the utility of overloading?
    Overloading allows having multiple functions with the same name, that do the same things or different things, but for different data types (even ones we created).

26. What is overriding? (supradefinirea)
    The ability to have two or more functions with the same name, one in the base class and one in the subclass, that differ through implementation.

27. What is the use of overriding?
    Allows us to have two or more functions with the same name, one in the base class and one in the subclass, that differ through implementation.

28. What is a virtual function?
    A member function that allows overriding the base class methods in the derived classes.
    It is used in the base class in order to ensure that the function is overridden (when a pointer of base class points to an object of a derived class)

    e.g.

```
class Base {
public:
        virtual void print() { cout << “Base function”  << endl; }
}
Class Derived : public Base {
Public:
        Void print() { cout << “Derived function” << endl; }
}
```

29. How is the concept of polymorphism implemented in C ++?
    overriding, overloading

30. What are the access modifiers and what visibility does it offer to the member data / functions in case of class derivation?
    3 access modifiers: PUBLIC, PRIVATE and PROTECTED.
    Public attributes and methods can be accessed in any part of the program.
    Protected attributes and methods can only be accessed in the subclasses.
    Private attributes and methods can only be accessed in the class they have been defined in.

31. In what context is the *private* access modifier useful?
    Private access modifiers are useful regarding DATA ENCAPSULATION. These don't allow any other classes (even subclasses) to access the class attributes.

32. What are the types of constructors in a class and what is the role of each?
    3 types: DEFAULT, WITH PARAMETERS and COPY.

    Default constructor: creates an object by initializing the attributes with constant values, that are NOT received as parameters.

    Ctor with parameters: creates an object by initializing the attributes with values received as parameters from the caller.
    Copy ctor: assigns space to an object and initializes it with the values of an existing object.
    It has a default form that copies bit by bit the value of the existing object in the memory area of the newly created object;
    It is automatically called in all cases when an object is defined and initialized;

33. What is the difference between the role of the = operator and that of the copy constructor?
    The = operator is called for 2 existing objects.
    The copy constructor creates a new object as a copy of an existing object.

34. When is the copy constructor called?
    The compiler automatically calls the copy constructor to copy the values of the objects in the parameter list (if sent by value) to the function stack.
    The compiler automatically calls the copy constructor to copy the value of the object returned by the function (if returned by value) to the calling program stack.
    We use the copy constructor when we want to create a copy of an already existing object.

35. When is the = operator called?
    When we want to copy the values of an existing, source object to a destination object.

36. What is a *memory leak* ?
    A memory leak occurs when we create a memory in heap but then forget to delete it.

37. What is a *dangling pointer*?
    A pointer that points to a memory zone (address) that has already been deleted.

38. What is the role of the destructor?
    The destructor is a public method (a member function) which destructs or deletes an object.

39. When is the destructor called?
    Destructors are called implicitly when an object needs to be destroyed.
    A destructor function is called automatically when the object goes out of scope:
    (1) the function ends
    (2) the program ends
    (3) a block containing local variables ends
    (4) a delete operator is called
    (5) at the end of the code block in which the object has been declared

40. What is HEAP memory?
    HEAP memory is allocated during the execution of instruction written by programmers.
    The allocation and deallocation are done manually, by the programmer, and the memory is allocated in any random order.

41. How do you allocate memory space in HEAP?
    Using the **new** operator.

42. How is memory freed in HEAP?
    Using the **delete** operator.

43. How is a *memory leak* generated?
    When an object gets destroyed/deleted, but its allocated heap memory is not freed.

44. What is the role of accessor functions in the class?
    They allow **read / write access** to the private attributes of the class;
    They implement the validation of the input data.

45. What is the role of friend functions in classes and what are their characteristics?
    Friend functions allow access to the private or protected area outside of the class.
    They do not have the *this* pointers.

46. What is the concept of encapsulation?
    Data is sealed/hidden inside the object and it can be accessed only by its methods/interface

47. What is the difference between overriding and overloading?
    OVERRIDING – defined functions with the same header in the same class or in the derived class. Overloading requires a hierarchy
    OVERLOADING – functions with the same header, that do the same thing or different things, for different data types. Can be used without a class-hierarchy.

48. How can the size of a char vector (static vs dynamic) be found?
    Strlen()

49. What is the default visibility within a class?
    Private

50. List three ways to use the * operator in C++.
    1. Multiplication
    2. Pointer declaration
    3. Extracting a value from an address

51. Why are accessor functions generally used instead of making the entire class public?
    (encapsulation) To control who has access/who can modify the class members

52. What type (class) is the object cin?
    Istream

53. What type (class) is the cout object?
    Ostream

54. What is the STL the abbreviation?
    Standard Template Library

55. Give an example of an STL container.
    Array, vector, list, set, map, stack, queue

56. Give an example of an STL algorithm.
    Copy, for_each, sort, find, transform

57. By what types of methods can an operator be overloaded?
    By class member functions o by a function outside the class

58. By what types of methods can the + for (object + int) operator be overloaded?
    By class or independent functions

59. By what types of methods can the + operator for (int + object) be overloaded?
    Independent function

60. When should "friend" be used in the context of overloading operators?
    When we want to access private attributes without accessor methods.

61. What does the copy constructor do?
    Creates a new object as a copy of an already existing object.

62. What is the name of the destructor?
    ~Class_Name()

63. What is the difference between a variable and an attribute?
    An attribute is a class field.

64. In what context is the protected access modifier useful?
    Inheritance – we use protected when we want only the derived classes to access the base class attributes, and not the whole program.

65. What is the difference between class and struct specifiers?
    Class – by default private attributes, can contain functions
    Struct – by default public, cannot contain functions

66. What is a class?
    Class – user-defined data type.

67. What is an object?
    An instance of a class.

68. What is a class instance?
    An object.

69. What is a constructor?
    A special function that creates space for the object internal state and initializes it with attributes and other logic operators.

70. What is the term *up-casting*?
    Allows us to implicitly convert object or pointers of the derived type into objects or pointers of the base type.

71. What is the term *down-casting*?
    Allows us to use an object or a pointer of the derived class to handle the base class.

72. What is a class framework?
    The hierarchy of a class.

73. Can constant methods be defined?
    No.

74. What is the order of execution of the constructors within the class hierarchies?
    Base constructor –> subclass constructor

75. What is the order of execution of the destructors within the class hierarchies?
    Subclass destructor –> base destructor

76. What role do virtual functions play in class hierarchies?
    Allow overriding the base class methods in the derived classes.

77. What is a pure virtual function?
    A function without a body, that will be overridden.

78. What is an abstract class?
    Classes that contain at least a pure virtual function.
    They have an interface role for classes that need to define a set of common methods

79. What restrictions does an abstract class impose?
    You can't instantiate abstract classes (they are used as support for inheritance).

80. How is multiple inheritance achieved in C ++?
    When a subclass has 2 or more base classes.

81. What are inline functions?
    Short functions that are not called, that allow quick code execution by avoiding the effort required for a function call.
    Default methods whose body is defined in the class are considered inline.

82. What is the *"is a"* concept? Exemplify
    Based on INHERITANCE -> subclass is a base class

    e.g.: Apple is a Fruit, Car is a Vehicle, House is a Building.

83. What is the *"has a"* concept? Examples
    Refers to the use of instance variables that are references to other objects.

    e.g.: Email has a Subject, Student has a Grade, Table has a Column

84. What is virtual inheritance?
    Virtual inheritance is used when we are dealing with multiple inheritance but want to prevent multiple instances of same class appearing in inheritance hierarchy.

85. At what point does overriding actually take place?
    During run-time.

86. At what point does the overloading actually take place?
    During compilation.

87. What is the default visibility within a structure?
    Public

88. How can a type 1:M relationship be modeled between two classes?

89. How can a type 1:1 relationship be modeled between two classes?

90. How can you do a base to sub-class cast?
    Dynamic_class<>()

91. How do we solve the problem of deadly diamond of death in the context of multiple inheritance?
    Avoid multiple inheritance

92. In what order are the constructors called in case of a multiple inheritance?
    In the order of the derivation
    Base -> derivate1 -> derivate2 …

93. How is the problem of a template method not working on a certain type of date?
    Using *auto* or constant attributes.

94. How do you define a pointer that can manage any type of memory area?
    Char*

95. What is the term *late-binding*?
    Virtual functions: the version of the function is determined at the time of execution.

96. What is the term *early-binding*?
97. Virtual functions: the version of the function is determined at the time of compilation.

98. What is the utility of virtual functions?
    - Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.
    - They are mainly used to achieve Runtime polymorphism

99. What can abstract classes be used for?
    Used as support for inheritance.

100. Can a pointer to an abstract class be used to handle objects such as derived classes?
    Yes

101. What is the usefulness of constant attributes?
    We can define attributes that won't be modified after initialization.

102. How do you define a class method that does not receive the *this* pointer?
    Static

1) What is the correct statement regarding next sequence ?

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

class Base
{
public:
    int x, y;
public:
    Base(int i, int j) { x = i; y = j; }
};

class Derived : public Base
{
public:
    Derived(int i, int j){
        this->x = i;
        this->y = j;
    }
    void print() { cout << x << " " << y; }
};

int main()
{
    Derived q(10, 10);
    q.print();
}
```

a. Generates a runtime error

b. Compiler error on Derived constructor !

c. It prints 10 10

d. Compiler error on lines 16 and 17 because x and y are not defined in Derived

e. Compiler error on lines 16 and 17 because x and y are not defined in Derived

1) Select the correct output regarding the next sequence

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

int main()
{
    try
    {
        throw 'a';
    }
    catch (int param) {
        cout << endl << "int exceptionn";
    }
    catch (...) {
        cout << endl << "default exceptionn";
    }
    cout << endl << "After Exception";
}
```

a.

int exception

After Exception

b. runtime error because the exception is not caught

c.

int exception

d.

default exception

After Exception !

e.

default exception

1) What is the output of the next sequence ?

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

class ClassA {
public:
    ClassA(){}
    virtual void print() {
        cout << endl << "Print A";
    }
    void publish() {
        cout << endl << "Publish A";
    }
};

class ClassB: public ClassA {
public:
    void print() {
        cout << endl << "Print B";
    }
    void publish() {
        cout << endl << "Publish B";
    }
};

int main() {

    ClassB b;

    ClassA a = b;
    ClassA *pa = &b;

    a.print();
    a.publish();
    pa->print();
    pa->publish();
```

a. Compiler error

b. Print B

Publish B

Print B

Publish B

c. Print B

Publish B

Print B

Publish A

d. Print A

Publish A

Print B

Publish A

e. Print A

Publish A

Print B

Publish B

f. Print B

Publish A

Print B

Publish A

1) What is the correct statement about the next sequence ?

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

class Base {
public:
    void fun() { cout << "Base fun!"; }
};

class Derived : public Base {
public:
    void fun() { cout << "Derived fun!"; }
};

int main() {
    Derived d;
    d.Base::fun();
}
```

a. Compiler error on line 16

b. Prints Base fun!

c. Compiler error on line 11

d. Prints Derived fun!

1) For the next code sequence indicate how many times you will see the "Class A constructor" message ?

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

class ClassA {
public:
    ClassA(){
        cout << endl << "Class A constructor";
    }
};

class ClassB {
    static ClassA A;
public:
    ClassB() {
    }
};
ClassA ClassB::A = ClassA();

int main() {

    ClassA a1;

    ClassB b1;
    ClassB b2;
}
```

a. 4 times

b. 1 time

c. 2 times

d. 3 times

e. You can't because the sequence has compiler errors

2) For the next code sequence, indicate how many times the ClassA destructor is called until the line 29 (the one with the end) is reached ?

```
4  #include<iostream>
5  using namespace std;
6
7  class ClassA {
8  public:
9      ClassA(){}
10     ~ClassA() { cout << endl << "Class Destructor"; }
11 };
12
13 class ClassB: public ClassA {
14 public:
15     void doSomething() {
16         ClassA a;
17         cout << endl << "Nothing";
18     }
19 };
20
21 int main() {
22     ClassA *pa = new ClassA();
23     ClassB b;
24
25     b.doSomething();
26     b.doSomething();
27     pa = new ClassA();
28
29     cout << endl << "The end";
30 }
```

Alegeți o opțiune:

a. The ClassA destructor is not called until that line

b. 3 times

c. 2 times

d. 4 times

e. 1 time

1) What will print the next sequence ?

```cpp
#include<iostream>

using namespace std;
class Base1 {
 public:
     Base1()
     { cout << " Base1 ctor" << endl;  }
};

class Base2 {
 public:
     Base2()
     { cout << "Base2 ctor" << endl;  }
};

class Derived: public Base1, public Base2 {
   public:
     Derived()
     {  cout << "Derived ctor" << endl;  }
};

int main()
{
    Derived d;

}
```

a. Compiler Error

b. Derived ctor

c. Base2 ctor

Derived ctor

d. Depends on the compiler

e. Base2 ctor

Base1 ctor

Derived ctor

f. Base1 ctor

Derived ctor

g. Base1 ctor

Base2 ctor

Derived ctor

1) What is the correct statement about the next sequence ?

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

class Base{
public:
    virtual void print(){
        cout << endl <<  "This is Base";
    }
};

class Derived : public Base{
public:
    virtual void print(){
        cout << endl << "This is Derived";
    }
};

void printInfo(Base p){
    p.print();
}

int main()
{
    Base b;
    Derived d;
    printInfo(b);
    printInfo(d);
}
```

a. This is Derived

This is Derived

b. This is Base

This is Derived

c. It generates a Compiler error in main()

d. This is Derived

This is Base

e. This is Base

This is Base

1) The output of the next code sequence is ?

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

class ClassA {
public:
    ClassA() { cout << endl << "Normal Constructor"; }
    ClassA(const ClassA &a) { cout << endl << "Copy constructor"; }
};

int main()
{
    ClassA *t1, *t2;
    t1 = new ClassA();
    t2 = new ClassA(*t1);
    ClassA t3 = *t1;
    ClassA t4;
    t4 = t3;
}
```

a.
Normal Constructor called
Copy Constructor called
Copy Constructor called
Normal Constructor called
Copy Constructor called

b. Normal Constructor
Copy Constructor
Copy Constructor
Normal Constructor

c.
Normal Constructor
Copy Constructor
Copy Constructor
Normal Constructor

d. Normal Constructor
Normal Constructor
Normal Constructor
Copy Constructor
Copy Constructor
Normal Constructor
Copy Constructor

e. Compiler Error

1) What will print the next sequence ?

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

int main()
{
    int x = -1;
    try {
        cout << endl << "Inside try";
        if (x < 0) {
            throw x;
            cout << endl << "After throw";
        }
    }
    catch (int x) {
        cout << endl << "Exception Caught";
    }

    cout << endl << "After catch";
}
```

a.

Inside try

Exception Caught

b.

Inside try

After throw

After catch

c.

Inside try

Exception Caught

After catch

d.

Inside try

Exception Caught

After throw

After catch

3) What is the output of the next sequence ?

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

class Base {
private:
    int i = 0, j = 0;
};
class Derived : public Base {
public:
    void show() {
        cout << " i = " << i << "  j = " << j;
    }
};
int main(void) {
    Derived d;
    d.show();
}
```

a. Compiler Error because i and j are private in Base

b. Compiler Error because i and j are private in Base

c. i = 0 j = 0

d. Compiler Error because i and j are not inherited

e. Compiler Error because there is no default constructor in Base

1) What statement is true about the next sequence ?

```
1  #include<iostream>
2  using namespace std;
3
4  class Base{
5  public:
6      void show(){
7          cout<<" I am a Base ";
8      }
9  };
10
11 class Derived: public Base{
12 public:
13     int value = 5;
14     void show(){
15         cout<<"I am a Derived";
16     }
17 };
18
19 int main(void)
20 {
21     Base *bp;
22     Derived d;
23     bp = &d;
24     bp->show();
25     cout << bp->value;
26 }
```

a. Compiler error on line 24

b. Compiler error on line 23

c. Compiler error on line 25

d. Prints:

I am a Base 5

e. Prints:

I am a Derived 5

1) Being given the next class definitions, indicate the ClassB attributes which MUST be initialized using the constructor initializer/initialization list (select 1 ore more answers)

```
1   #include<iostream>
2   using namespace std;
3
4   class ClassA {
5       int x, y;
6   public:
7       ClassA(int i, int j): x(i), y(j){}
8   };
9
10  class ClassB {
11      const int z;
12      int w;
13      ClassA a;
14      ClassA *pa;
15  public:
16      ClassB() {
17          //to be completed
18      }
19  };
20
21  int main() {
22
23  }
```

a. a

b. pa

c. You can't use the constructor initialization list for any of the attributes

d. z

e. w

1) If you have the next sequence and you want to write the instruction on line 13, what is the correct form of the overloaded operator + that you need to implement ?

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

class B {
    int x;
public:
    B(int value) : x(value) {}
};

int main() {
    B b1(10), b2(20);

    b1 = 100 + b2;
}
```

a. As a global method: B operator+(int value, B b)

b. As a class member function method: B operator+(int value, B b)

c. As a class member function method: B operator+(B b) NU

d. As a global method: B operator+(B b, int value)

e. As a class member function method: B operator+(int value)

1) What is the output of following program?

```
#include<iostream>
using namespace std;
class ClassA {
    ClassA() { cout << "Constructor called"; }
};

int main()
{
    ClassA a1;
}
```

a. The object is not created because it has no attributes

b. Compiler Error

c. Runtime Error

d. Constructor called

1) What statement is true about the next sequence ?

```
1   #include<iostream>
2   using namespace std;
3
4   class Base {};
5
6   class Derived: public Base {};
7
8   int main()
9   {
10      Base *bp = new Derived();
11      Derived *dp = new Base();
12  }
```

Alegeți o opțiune:

a. Runtime Error

b. Compiler error on line 10

c. Compiler error on line 11

d. No Compiler error

2) What is DOWN-casting?

a. convert an object to a pointer type

b. convert a pointer type to an object type

c. converting subclass type pointers or objects to base type pointers or objects

d. converting base type pointers or objects to subclass type pointers or objects

2) What is the difference between the role of the operator = and copy constructor?

a. copy constructor creates a new object, the = operator works with two existing objects;

b. = operator creates a new object with the same values, copy constructor copies two existing objects;

c. there is no difference, both functions create a new object from another object;

d. operator = deletes the memory space for the object name;

1) How does C++ compiler differs between overloaded postfix and prefix operators?

Alegeți o opțiune:

a. C++ doesn't allow both operators to be overlaoded in a class

b. A postfix ++ has a dummy parameter

c. By making prefix ++ as a global function and postfix as a member function

d. A prefix ++ has a dummy parameter

1) Which of the following are not inherited?

a. public data members

b. constructors and destructor

c. accessor methods

d. protected attributes

e. private attributes

4) A method in a derived/subclass class that has the same signature as another method in the parent/base class:

a. will generate an error message

b. It will be executed only immediately after the base class method execution will finish

c. will override the base class method

d. It will be overwritten by the method in the base class

e. none of the presented affirmations is true

1) When the inheritance is public, the private methods in base class are _____ in the derived class (in C++).

a. inaccessible

b. accessible

c. protected

d. public

5) What is the concept of memory leaks in C++:

a. This concept does not exist in C++

b. Incorrect initialization of an object

c. Wrong definition of the copy constructor

d. Deleting storage space in HEAP

e. Incorrect initialization of a pointer

f. Assigning a HEAP memory space that is no longer referenced by any pointer

1) What is THIS inside a C ++ class constructor:

a. Pointer that manages the built object address

b. Optional variable that can be used to indicate which variables are attributes and which are not

c. You can not use THIS in the constructor

d. Pointer which manages the destroyed object address

e. Value of the object which is calling the constructor method

Catch blocks must:

a. be used in any object oriented program

b. be used immediately after the try block

c. be used inside the try block

d. be used immediately after calling throw

e. none of these affirmations is true

6) What is the role of accessor methods in a class?

a. provide access ONLY to constant attributes of the class;

b. restrict access to the data members of the class;

c. restrict access to class static attributes;

d. provide access in read / write mode to private class attributes;

7) What is the significance of the friend keyword in C ++ ?

a. It is used to give access to global methods or other classes on the current class private area

b. It doesn't exist in C++

c. It is used to define global methods

d. It is used to overload operators

e. It is used to indicate that a function does not return values

2) When a copy constructor may be called? (select 1 or more answers)

a. When you define a static array of objects

b. When an object of the class is returned by value from a function

c. When you define a dynamic array of objects

d. When an object of the class is passed (to a function) by value as an argument

e. When an object is constructed based on another object of the same class

f. ……..ClassType obj2 = obj1 pattern

What represents the “this” keyword in C++?

Is the address o the object that calls a class member function

For an int array [] with the values {1,2,3,4} what value will display cout << *(array + 2) ?

3

What is the effect of the next statement in int * address (int*)25?

Copies the value 25 into the pointer variable

Given the next Text class, what is the size of in bytes of a Test object?

Other value

When you define a class, what is the default C++ modifier that you get?

private

1)Which of the following CANNOT be passed to a function in C++ ?

a. Array

b. Constant

c. Header file

d. Structure

e. Pointer to a variable

2) Assume a x86 processor. Also, assume that there is no alignment in objects. Predict the output following program.

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

class Test
{
    static int x;
    int *ptr;
    int y;
};

int main()
{
    Test t;
    cout << sizeof(t) << " ";
    cout << sizeof(Test *);
}
```

Alegeți o opțiune:

a. 12 12

b. 8 4

c. 8 8

d. 12 8

e. 12 4

3) An object is _____ of a class.

a. a member function

b. an encapsulation

c. an interface

d. an instance

4) Which of the following(one or more answers) is/are NOT part of an object internal structure?

a**. Static variable**

b. **Private static variable**

c. **Static constant variable**

d. Private variable (any type and not static)

e. Public static variable

5) Predict the output of following C++ program

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

class Empty {};

int main()
{
  cout << sizeof(Empty);
  return 0;
}
```

a. Runtime Error

b. Compiler error

c. 0

d. A non-zero value

6) Being given the next class and main() implementation, please say how many times the destructor is called before reaching line 16 ?

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

class Test {
    int attr1;
    char attr2;
};

int main() {
    Test tObject;
    Test values[100];
    Test* newValues = new Test[10];

    delete[] newValues;

    cout << endl << "Phase 1 end";
}
```

a. We don't have a default destructor

b. 10

c. 100

d. 0

e. 111

7) What is a "dangling pointer" in C++ ?

a. A normal pointer

b. A pointer initialized with nullptr

c. A pointer initialized with NULL

d. A pointer that stores a valid address in HEAP

e. A pointer initialized with 0XCCCC....CC

f. A pointer that stores an address from HEAP that does not belong anymore to your processor

8) Being given the next class and main() implementation, please say how many times the default constructor is called before reaching line 16 ?

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

class Test {
    int attr1;
    char attr2;
};

int main() {
    Test tObject;
    Test values[100];
    Test* newValues = new Test[10];

    delete[] newValues;

    cout << endl << "Phase 1 end";
}
```

Alegeți o opțiune:

a. 10

b. 100

c. We don't have a default constructor

d. 111

e. 1

9) Being given the next class Test, what will be the output/result of compiling and running the sequence

```cpp
class Test {
    int x;
};
int main()
{
  Test t;
  cout << t.x;
  return 0;
}
```

a. Garbage/Residual value

b. Runtime Error

c. 0

d. Compiler Error

10) For a class Student what is the "this" variable type ?

Alegeți o opțiune:

a. a float

b. an integer

c. Student

d. Student*

e. char*

11) A class member function can always access the data in _____ , (in C++).

a. the object of which it is a member

b. the public part of its class

c. the private part of its class

d. the class of which it is member

12) When one object reference/pointer variable is assigned to another object reference/pointer variable then

a. it is illegal to assign one object reference/pointer variable to another object reference/pointer variable.

b. a copy of the reference is created. You have only one object

c. a copy of the reference is not created. You have 2 different objects with the same values referenced by 2 different pointers

d. a copy of the object is created. You have 2 different objects with the same values

13) What defines the public interface of a class definition ?

a. collection of public methods

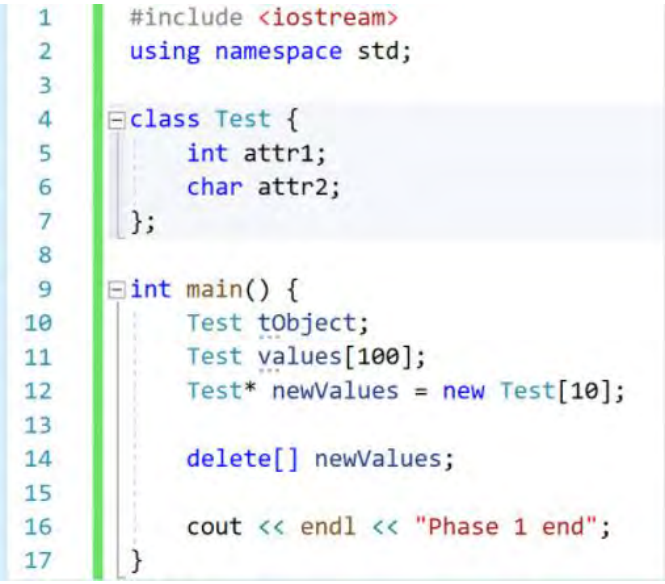
b. collection of public and private attributes

c. only the collection of public attributes

d. only the collection of public static methods

e. the class description

15) Being given the next class and main() implementation, please say how many times the copy constructor is called before reaching line 16 ?



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

class Test {
    int attr1;
    char attr2;
};

int main() {
    Test tObject;
    Test values[100];
    Test* newValues = new Test[10];

    delete[] newValues;

    cout << endl << "Phase 1 end";
}
```

a. 111

b. 0

c. 100

d. We don't have a default copy constructor

e. 10

16) When is the copy constructor called (choose one or more answers) ?

a. When you pass an object by reference to a function call

b. When you pass an object by pointer to a function call poate

c. When you create a new object based on existing one using ClassType obj2 = obj1 pattern

d. When you receive an object by reference from a function call (the function returns the object reference)

e. When you pass an object by value to a function call

f. When you receive an object by value from a function call (the function returns the object)

17) If we have (check the given class Card) we can use _____ to access int s:

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

class Card {
public:
   int s;
};

int main() {
 Card a;
 Card* p = &a;
}
```

a. p->int

b. p->s

c. a->s

d. a.int

e. p.s

18) What is the console output of the following program?

```
#include <iostream>
using namespace std;
main() {
   char s[] = "hello", t[] = "hello";
   if(s==t)
      cout<<"equal strings";
}
```

a. Compilation error

b. unequal strings

c. equal strings

d. no output

19) If you have a x64 application (for a x64 processor) and the next definition double* values = new double[10], what is the size in bytes of values ?

a. 40

b. 8

c. 80

d. 4

e. 10

20) What is the difference between struct and class in C++?

a. All members of a structure are public and structures don't have constructors and destructors

b. Members of a class are private by default and members of struct are public by default.

c. All of statements are true

d. All members of a structure are public and structures don't have copy constructors

1) Being given the next code sequence, select all answers (one or more) which are correct statements regarding the number of created objects (from the 3 classes):

```
class Grade {
    int value;
};

class GradeBook {
    Grade grades[50];
    int noGrades = 0;
};

class Student {
    GradeBook gradeBook;
};

int main() {
    Student student;
}
```

a. 0 Student objects are created

b. 1 Student object is created

c. 50 Grade objects are created

d. 0 GradeBook objects are created

e. 1 GradeBook object is created

f. 0 Grade objects are created

g. 1 Grade object is created

2) Being given the next code sequence, select all statements (one or more) which are true regarding the called constructors

```
class Grade {
    int value;
};

class GradeBook {
    Grade grades[50];
    int noGrades = 0;
    string uniqueId;
public:
    GradeBook() {}
    GradeBook(string id):uniqueId(id) {}
};

class Student {
    GradeBook gradeBook;
    string name = "";
public:
    Student() {}
    Student(string Name, string Id):gradeBook(Id), name(Name) {}
};

int main() {
    Student student;
}
```

a. The Grade default constructor is called

b. The GradeBook constructor with arguments is called

c. The Student default constructor is called

d. The Student constructor with arguments is called

e. The GradeBook default constructor is called

3) Being given the next code sequence, select all statements (one or more) which are true regarding the called constructors ("John"…)

```
class Grade {
    int value;
};

class GradeBook {
    Grade grades[50];
    int noGrades = 0;
    string uniqueId;
public:
    GradeBook() {}
    GradeBook(string id):uniqueId(id) {}
};

class Student {
    GradeBook gradeBook;
    string name = "";
public:
    Student() {}
    Student(string Name, string Id):gradeBook(Id), name(Name) {}
};

int main() {
    Student student("John", "GB-1001");
}
```

a. The GradeBook default constructor is called

b. The Student default constructor is called

c. The GradeBook constructor with arguments is called

d. The Grade default constructor is called

e. The Student constructor with arguments is called

4) Being given the next code sequence, select all answers (one or more) which are correct statements regarding the number of created objects (from the 3 classes): Gradebook fara *

```
class Grade {
    int value;
};

class GradeBook {
    Grade* grades;
    int noGrades = 0;
};

class Student {
    GradeBook gradeBook;
};

int main() {
    Student student;
}
```

Alegeți una sau mai multe opțiuni: (Grade cu *, Gradebook fara)

a. 1 Student object is created

b. 1 Grade object is created sau asta

c. 0 Grade objects are created

d. 0 Student objects are created

e. 0 GradeBook objects are created

f. 50 Grade objects are created sau asta

g. 1 GradeBook object is created

5) Being given the next code sequence, select all answers (one or more) which are correct statements regarding the number of created objects (from the 3 classes): (GradeBook* gradebook)

```
class Grade {
    int value;
};

class GradeBook {
    Grade* grades;
    int noGrades = 0;
};

class Student {
    GradeBook* gradeBook;
};

int main() {
    Student student;
}
```

Alegeți una sau mai multe opțiuni: (Grade cu *, Gradebook cu *)

a. 1 Student object is created

b. 1 Grade object is created

c. 0 Student objects are created

d. 0 Grade objects are created

e. 1 GradeBook object is created

f. 50 Grade objects are created

g. 0 GradeBook objects are created

1) Being given the next sequence, select the output generated by it

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

class A {
public:
    ~A() { cout << endl << "A destructor"; }
};

class B: public A {
public:
    ~B() { cout << endl << "B destructor"; }
};

class C : public B {
public:
    ~C() { cout << endl << "C destructor"; }
};

void main() {
    A* pAObject = new A();
    A aObject;
    C cObject;
}
```

a. C destructor
B destructor
A destructor
b. C destructor
B destructor
A destructor
A destructor
A destructor
c. C destructor
A destructor
A destructor
d. C destructor
A destructor
e. **C destructor**
B destructor
A destructor
A destructor

2) Assume that an integer takes 4 bytes and the size of each object is exactly the size of its attributes (no alignment), what will be the output.

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

class base {
    int arr[10];
};

class b1 : public base { };

class b2 : public base { };

class derived : public b1, public b2 {};

int main(void)
{
    cout << sizeof(derived);
}
```

a. 0

b. 40

c. 4

d. 80

e. 8

3) For inheritance, what is the order of execution for constructors when you create a Subclass class object ? And for destructors ? (select 1 ore more answers)

a. Subclass constructor, Base class constructor

b. Base class destructor, Subclass destructor

c. Only the Subclass destructor

d. Base class constructor, Subclass constructor

e. Only the Subclass constructor

f. Subclass destructor, Base class destructor

4) Assume that it is an app for a x86 processor and the size of each object is exactly the size of its attributes (no alignment), what will be the output.

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

class base {
    int arr[10];
};

class subclass : public base {
    char text[10];
    int* pointer = new int[5];
};


int main(void)
{
    cout << sizeof(subclass);
}
```

a. 74

b. 64

c. 70

d. 54

e. 0

5) Being given the next sequence, what it will print ?

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

class P {
public:
    void print() { cout << " Inside P"; }
};

class Q : public P {
public:
    void print() { cout << " Inside Q"; }
};

class R : public Q { };

int main(void)
{
    R r;
    r.print();
}
```

a. Inside P

b. Compiler Error: Ambiguous call to print()

c. Inside Q

d. Compiler Error on line 11: You can't redefine a base class method

3) Fill up the next sequence. Constructors have _____ implicit return type.

a. void

b. int

c. its class type

d. void*

e. char

4) A method in a derived/subclass class that has the same signature as another method in the parent/base class:

a. will generate an error message

b. It will be executed only immediately after the base class method execution will finish

c. will override the base class method

d. It will be overwritten by the method in the base class

e. none of the presented affirmations is true

5) In terms of pseudo-code with what sequence of operators is the following statement equivalent:

a1 = 10 += a2;

a. a1.operator=(operator+=(a2,10));

b. a1.operator=(operator+=(10,a2))

c. operator=(a1,a2.operator+=(10));

d. a1.operator=(a2.operator+=(10));

Being given the next source code (the 3 called fuctions are defined correctly) indicate what it will print, if **function1() executes throw MyException();**

```
int main() {
	int vb = 10;
	try {
		vb = 20;
```

```
                function1();
                vb = 30;
                function2();
                vb = 40;
                function3();
                vb = 50;
        }
        catch (MyException Ex) {
                vb = 60;
        }
        catch (MyException* pEx) {
        }
        catch (int ex) {
                vb = ex;
        }
        catch (...) {
                vb = 80;
        }
        cout << endl << vb;

}
```

60

Being given the next source code (the 3 called fuctions are defined correctly) indicate what it will print, if neither the 3 functions is not throwing anything?

Nu 10

50?

Being given the next source code (the 3 called functions are defined correctly) indicate what it will print, **if function 3 will execute throw 70;**

70

Being given the next source code (the 3 called fuctions are defined correctly) indicate what it will print, if **function2() will execute throw new MyException();**

Nu 80

60?

Being given the next sequence, what it will print ? q q q.print(); inside

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

class P {
public:
    void print() { cout << " Inside P"; }
};

class Q : public P {
public:
    void print() { cout << " Inside Q"; }
};

int main(void)
{
    Q q;
    q.print();
}
```

Alegeți o opțiune:

a. Compiler Error on line 11: You can't redefine a base class method

b. Inside Q

c. Compiler Error: Ambiguous call to print()

d. Inside P

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

class A {
public:
    A() { cout << endl << "A ctor call"; }
};

class B: public A {
public:
    B() { cout << endl << "B ctor call"; }
};

class C : private B {
public:
    C() { cout << endl << "C ctor call"; }
};

void main() {
    A* pAObject = new A();
    C* pCObject = nullptr;
    C cObject;
}
```

a. A ctor call

C ctor call
B ctor call
A ctor call

b. A ctor call

A ctor call
B ctor call
C ctor call

c.
A ctor call

A ctor call
C ctor call

d. A ctor call
A ctor call
B ctor call
C ctor call
A ctor call
B ctor call
C ctor call

e. The sequence will generate a compiler error as C is privately inheriting B, so it will not be able to call the B constructor

f.
A ctor call
C ctor call

1) The polymorphism concept is implemented in OOP by ? (choose one or more answers)

Alegeți una sau mai multe opțiuni:

a. By explicitly calling the base class constructor

b. Overriding methods inherited from the base class

c. Overloading operators

d. By extending a class into a subclass

e. Overloading methods from the same class

What is the output of the following program? Const a++

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

main() {
    int const a = 5;

    a++;
    cout<<a;
}
```

Alegeți o opțiune:

- a. Runtime error
- b. 5
- c. 6
- **d. Compile error**

When you define a class, what is the default C++ access modifier that you get ?

- a. default
- b. public
- **c. private**
- d. there is none. you need to explicitly define it
- e. inaccessible
- f. protected

What represents the "this" keyword in C++ ? Select one or more answers

- a. Is the address of the object destroyed in the constructor
- **b. Is the address of the object created in the constructor**
- **c. Is the value o the object that calls a class member function**
- **d. Is the address o the object that calls a class member function**
- e. There is no this keyword

Given the next Test class, what attributes can you access directly from main() for a Test object ? Select one or more answers attr1

```
class Test {
    int attr1;
    unsigned int attr2;
    bool attr3;
public:
    char* attr4;
private:
    double attr5[10];
};
```

- [ ] a. attr1
- [ ] b. attr3
- [ ] c. attr5
- [x] **d. attr4**
- [ ] e. attr2

For int * array = new int [100]; the size of the array variable is (in bytes and on a x86 platform))

Alegeți o opțiune:

- (•) **a. 4**

The overloading concept can be applied in C++ for (select one or more answers)

Class member functions

Constructors

Operators

What will output the following program?

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

class Car {
public:
        int year
};

int main() {
```

```
        Car c;
        Car c2 = c;
        cout << c2.year;
}
```

Compilation error

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

long square(int &x) {
        x = x * x;
        return x;
}

int main() {
        int x = 5;
        cout << x << endl;
        square(x);
        cout << x << endl;
}
```

5

25