Duration: 2 hours _____ .

Name: _____ Date: _____ .
Allowance: Open Book Exam.

Object Oriented Programming - Sample Exam

Q.1 True or False:

1. If the class **`Y`** is a sub-class of the class **`X`**, then an object of the class **`Y`** will only inherit the protected members declared in the class **`X`** . **(F)**
2. A **`public`** member of a class can not be directly accessed by its name in the main() function. **(F)**
3. A friend function is allowed to access the private members of a class, but not the protected members of the class. **(F)**
4. If a programmer does not declare a constructor for a certain class, then the compiler will automatically create a default constructor for that class. **(True)**
5. In order to access a static variable of a class we have to write the class name before the variable name. **(True)**
6. The ***this*** pointer is a pointer that points to the object that the method has been called for. **(True)**
7. Function overloading is considered as one of the ways of polymorphism in object oriented programming. **(True)**
8. The constructor of a class is a member function that returns void. **(F)**
9. Encapsulation means extending a class to increase more member functions and member variables in a derived class. **(F)**
10. When a certain value is assigned to a static member variable at some point of the program, then it is not possible to assign a different value to this variable at a further point in the program. **(F)**
11. When we overload a function, we declare another function with the same name and we must specify a different number of parameters. **(F)**

Q.2 Choose the correct answer:

1. **A constructor is:**
   a) A member function that removes an object from the memory.
   b) A member function that is called automatically when an object is being removed from the memory.
   c) A member function that is used to initialize variables of a declared object.
   d) None of the above.

2. **Given the class below, what changes do we need to make to the class in order to have a facility to test if the values of two objects' data are equal?**

```
class MyClass
{
   private:
           float a ;
```

```
          float b ;
   public:
         MyClass( )

               a = 0 ;
               b = 0 ;
         }

         void setValues(int x , int y)
         {
               a = x ;
               b = y ;
         }
};
```

a) Declare a third temporary member integer.
b) Do operator = = overloading.
c) Create a new member function: **`void testEqual( )`**, that takes no parameters.
d) None of the above.

3. **Which of the following statements are true about destructors?**
a) Destructors are special functions with the same name of the class and are preceded by a tilde character (~).
b) Destructors cannot be overloaded.
c) A destructor has no return type.
d) All of the above.
e) None of the above.

4. **Which of the following most closely describes the process of overloading?**
a) A class with the same name replaces the functionality of a class defined earlier in the hierarchy.
b) A method with the same name completely replaces the functionality of a method defined earlier in the hierarchy.
c) A method with the same name but different parameters gives multiple uses for the same method name.
d) A class is prevented from accessing methods in its immediate ancestor.

5. **What will happen when you attempt to compile the code below?**

```
class MyClass
{
         int x ;

   public:
         int y ;
};

void main()
{
   MyClass obj ;
   obj.x = 100 ;
```

```
   obj.y = 500 ;
}
```

a) Compiler Error because **x** is not accessible.
b) Compiler Error because **y** is not accessible.
c) Compiler Error because **MyClass** is an abstract class and can not be instantiated.
d) a and **b**.
e) The code will compile successfully.

6. **'A plane is a machine that has a motor and has wings'.**
   'A refrigerator is a machine that has a motor and has shelves'.
   Which of the following best describes the previous statements as a set of classes?
   a) 3 classes: A **machine** class that has one attribute: *motor*. A **plane** class that inherits from the **machine** class. And a **refrigerator** class that inherits from the **plane** class.
   b) 3 classes: A **machine** class that has one attribute: *motor*. A **plane** class that inherits from the **machine** class. And a **refrigerator** class that also inherits from the **machine** class.
   c) 2 classes: A **plane** class that has two attributes**,** and a **refrigerator** class that also has two attributes.
   d) 1 class: A **machine** class that has an attribute for the *type of machine*.

7. **The following function prototype uses the default arguments feature:**
   `void myFunc(int x=3 , int y) ;`
   is the function valid this way?
   a) Yes, it is ok to give a default parameter for **x** and not for **y**.
   b) No, a function must always declare all its variables with default values.
   c) No, you must only give a default parameter for **y** but not for **x**.
   d) No, there should not be any arguments without default values to the right of the default arguments.

8. **A derived class can directly access:**
   a) The protected and public members of the base class.
   b) The private and protected members of the base class.
   c) Only the protected members of the base class.
   d) Only the private members of the base class.
   e) The public, private, and protected members of the base class.

9. **Assume you have a class `X` that contains an object of class `Y`. Assume that we declare an object of class `X` in the main( ) function. When will the body of the constructor of class `Y` be executed?**
   a) When any member function of the class **`X`** is called.
   b) After the body of the constructor of class **`X`** is executed.
   c) Before the body of the constructor of class **`X`** is executed.
   d) We can not determine exactly when it will be executed.

10. **An embedded (aggregated) object is:**
   a) An object that s declared with the default constructor.
   b) An object that is included by another object.

c) An object from a class that is inherited from another class.
d) b and **c**.
e) None of the above.

Q.3 Answer the following question:

Using your own words, explain the following in brief:
(i.e. discuss in a few lines or a short paragraph…. You don't need to write too much!):

A) What is the difference between **Multi-Level Inheritance** and **Multiple Inheritance**?

B) What are the core features of Object Oriented Programming?

C) What is the meaning of aggregation relation? What is the effect of such relation on the sequence of execution of the constructors?

D) What are the characteristics of a constructor?

Q.4 Answer the following question:

Design the set of classes that would best represent the following situation:

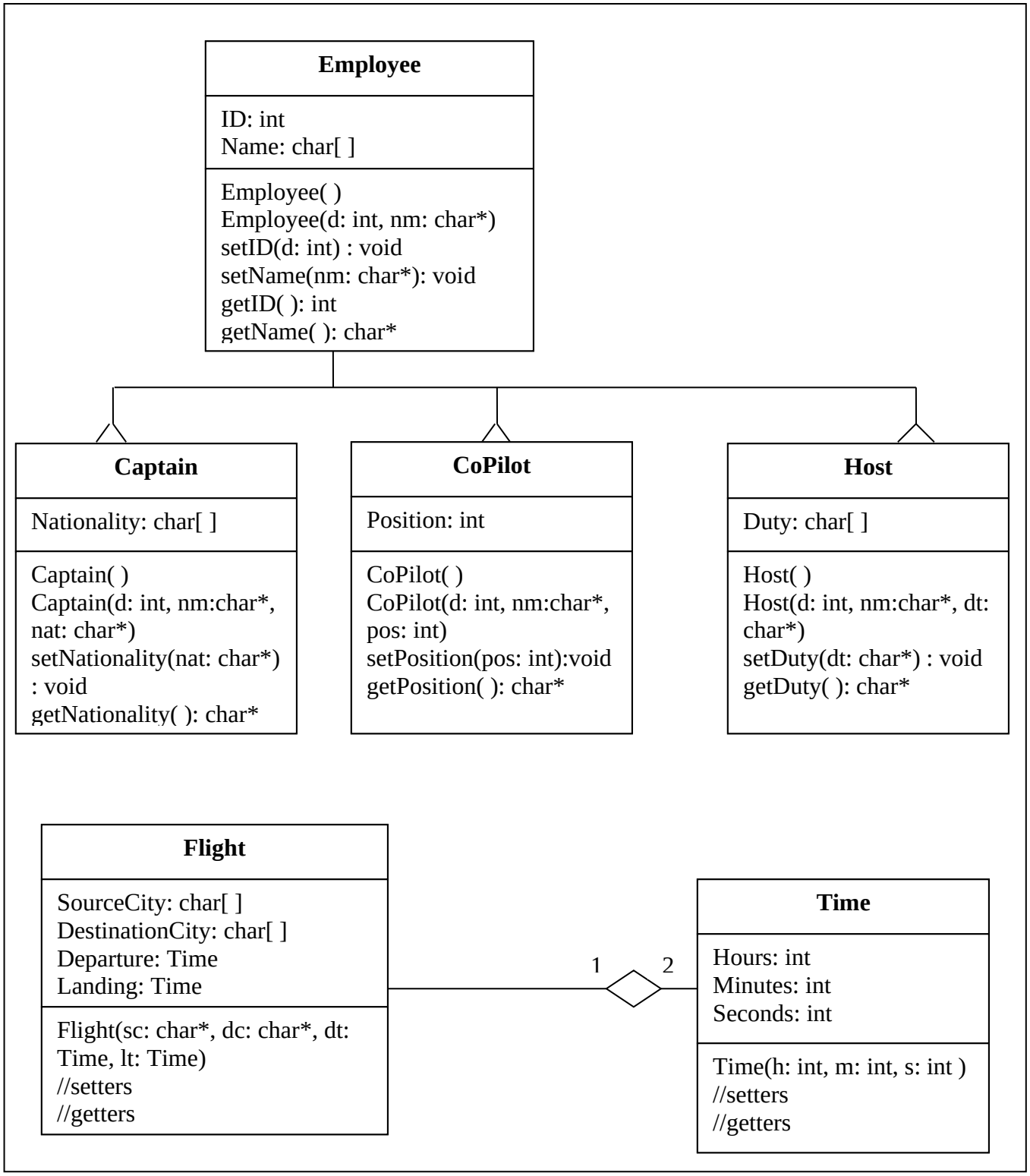
(Note: ***Design*** *means*: you can either draw a **UML diagram**. **OR** write the **class body**.
Whichever way you choose, make sure that you show the **data types** of the **attributes**, and write the **prototypes** of the **constructors** and **methods**. **YOU DO NOT NEED** to write any code inside the body of the methods).

You are asked to design a set of classes for an Airlines Company information system. The company has many members of staff: Captains, Co-Pilots, and Hosts. The company also wants to store information about its flight journeys. The required information is shown in details below:

Captain: ID, Name, Nationality .
Co-Pilot: ID, Name, Position.
Host: ID, Name, DutyDetails.
Flight: SourceCity, DestinationCity, Time of departure, Time of landing.

Hint: You can make any additional assumptions when needed. If you do so, then write one sentence about your assumption.

Solution using a simple UML diagram: (either use UML or write the class, but not BOTH!)



Assumptions: We will assume that Time is a new datatype.

Other solution by writing the class with the function prototypes:

```
Class Employee
{
   private:
         int ID ;
         char name[50] ;
   public:
         Employee( ) ;
         Employee(int, char*) ;
         void setID(int) ;
         void setName(char*) ;
         int getID( ) ;
         char* getName( );
}

class Captain : public Employee
{
   private:
         char nationality[20] ;
   public:
         Captain( ) ;
         Captain(int, char*, char*) ;
         void setNationality(char*) ;
         char* getNationality( ) ;
}

class CoPilot : public Employee
{
   private:
         int position ;
   public:
         CoPilot( ) ;
         CoPilot(int, char*, int) ;
         void setPosition(int) ;
         int getPosition( ) ;
}

class Host : public Employee
{
   private:
         char duty[70] ;
   public:
         Host( ) ;
         Host(int, char*, char*) ;
         void setDuty(char*) ;
         char* getDuty( ) ;
}
```

```
class Time
{
   private:
         int hours ;
         int minutes ;
         int seconds ;

   public:
         Time(int, int, int) ;
         //setters
         //getters
}

class Flight
{
   private:
         char sourceCity[20] ;
         char destinationCity[20] ;
         Time departureTime ;
         Time landingTime ;
   public:
         Flight(char*, char*, Time, Time) ;
         //setters
         //getters
}
```

Good Luck

9) The "this" pointer is an [illegible]

10) When overloading a certain function, it is obligatory that we specify a differe[illegible] number of parameters for the new function.

Q.2- Choose the correct answer(s) (Choose all that apply) (50 Marks)

1) Which of the following is true about an object member function?

A) It can be called using the name of the class.
B) It can access static variables of the class.
C) It has a "this" pointer as an implicit parameter passed to it.
D) It can access the instance variables.
E) It cannot be overloaded.
F) It can call other member functions from inside it.

2) Which of the following is true about the function prototype below?

void myFunc (int myDef=17, int myVar , int myNormalVar=5) ;

A) We should also give a default value to myVar.

B) We must only give a default parameter for myNormalVar and not the others.
C) The function is correct in that way.

this → a
*this.a

3) class Super

```
{
    protected:
            Super(int a)
                {
                    this.a = a;     // Line 1
                }
    private:                        // Line 2
            int a;
};
class Sub: public Super
{
    public:
            Sub(int a):Super(a) { }
    public:
            Sub()               //Line 3
            {
                this.a= 5;      //Line 4
            }
};
```

Which steps will allow Sub to compile?

A) Class Sub compile successfully.
B) Comment Line 2
C) In Line 1 and 4 to, use (*this).a instead of this.a;
D) Change Line 1 and 4 to, this(a);
E) Change Line 3 to, Sub(): Super(5)
F) Change Line 3 to, Sub(): this(5)
G) All the above.

4) **What will be the output when you compile and run the following piece of code?**

```
class Test
    { int x;
     public:
        Test( ) { x = 0; }
        Test(int y) { x = y++; }
        Test(Test &r)  { x = ++r.x; }
        void print( )    {cout<<x ; }
        };
    void main()
        {   Test t(1) ;
            t.print() ;
```

1

```
        Test x(t) ;
        x.print() ;
        t.print() ;
    }
```

A) 121
B) 122
C) 222
D) 233

5)
```
class Parent
    { public:
            int x;
            Parent(int m)  { x = m ; }
    };
class Child : protected Parent
    {
        public:
            int y;
            Child(int m, int n) : Parent(m) {  y = n ;  }
    };
class GrandChild : public Child
    {   int z ;
        public:
            GrandChild(int a, int b, int c) : Child(a,b) {  z = c ; }
    };
void main( )
    {  GrandChild obj(3,5,7);
       cout<<"Value of x is: "<<obj.x <<endl ;          //Line 1
       cout<<"Value of y is: "<<obj.y <<endl ;          //Line 2
       cout<<"Value of z is: "<<obj.z <<endl ;          //Line 3
    }
```

A) Compiler Error at Line 1
B) Compiler Error at Line 2
C) Compiler Error at Line 3
D) The code compiles successfully.

6)What will be the output when you compile and run the following piece of code

```
class Parent
{
    int x;
    Parent(int m){ x = m ; }
};
class Child : public Parent
{ public:
```

```
        int y;
        Child(int m, int n) : Parent(m) //Line 1
        {  y = n ;  }
    };

void main( )
{
   Child obj(3,5,7);          //Line 2
   cout<<"Value of x is: "<<obj.x <<endl ;  //Line 3
   cout<<"Value of y is: "<<obj.y <<endl ;  //Line 4
}
```

A) Compiler Error at Line 1
B) Compiler Error at Line 2
C) Compiler Error at Line 3
D) The code compiles successfully.

7) In order to turn a class into an abstract class, which of the following do we need to do?

A) Write the abstract keyword before the name of the class.
B) Make the class a pure virtual class.
C) Write one or more pure virtual functions inside the class.
D) A and C.
E) None of the above

8) What will be the output when you compile and run the following piece of code?

```
class Parent private
        {  int y;
           static int z;
           public:
                    Parent( )
                        {
                          z=0;   // Line1
                        }
                    Parent (int a=5)   //Line 2
                        {
                          y=a;        y = 4
                        }
        };
void main( )
        {
           Parent d(4);     //Line 3
           Parent m;        //Line 4
        }
```

A) Compilation Error at Line 1, an object member function cannot access a static member
B) Compilation Error at Line 2, constructor should initialize static member (z=0;)
C) Compilation Error at Line 3
D) Compilation Error at Line 4.
E) The code compiles successfully.

9) What will be the output when you compile and run the following piece of code?

```
class Parent
  {
    protected:
            int x;
    public:
            Parent(int m)
                    { x = m ; }
    friend void display( );
  };
class Child : public Parent
  {
    private:
            int y;
    public:
          Child(int m, int n) : Parent(m)
          {  y = n ;  }
  };
void display ()
        {
          Child c(3,4);
          cout <<"x="<<c.x<<"y="<<c.y; // Line 1
        }

void main ()
        {
          display();
        }
```

A) Compilation Error at Line 1, Child::x is inaccessible
B) Compilation Error at Line 1, Child::y is inaccessible
C) A and B
D) The code compiles successfully.

10) What will be the output when you compile and run the following piece of code?

```
class Nice
    {  int a ;
        public:
            Nice( ) { a = 0 ; }
            Nice(Nice & myN)
                {  this →a =  myN.a ;
                   cout<<"I am the copy constructor\n" ;
                }
            void setA(int m)  { a = m; }
             int getA()   { return a ; }
    };
void show(Nice &obj)
{   cout<<"I am the show function, value is: " << obj.getA() ; }
void main()
{  Nice n1;
   n1.setA(15) ;
   show(n1) ;
}
```

A) I am the show function, value is: 15.
B) I am the show function, value is: 15.
   I am the copy constructor.
C) I am the copy constructor.
   I am the show function, value is: 15.
D) I am the copy constructor.

11) What will be the output when you compile and run the following piece of code?

```
class A
{
   int x;
   protected :   int y ;
   public   :
            A(int x1=5,int y1=3)   { x=x1;y=y1; }
            void M1( ) {cout <<"\n This is M1() in class A:Base class";}
            void M3( ) {cout <<"\n This is M3() in class A:Base class";}
```

```
};
class B:private A
    {   private
        int w; int x; int y; M1(); M3();
        protected:   int v;
        public  :
                B(int v1=3, int w1=9) { v=v1;w=w1; }
                void M3()
                 { M1( );  //Line 1     M1() this is
                   y++;    //Line2
                 }
    };
class C: public B
    {   public:
        void  M4( )
              {  M1( );   //Line 3     M1() this is
                 y++;     //Line 4
              }
    };
void main()                                  x=5, y=3, w=9, v=3
    { B b1;                                    y=4
      b1.M3( );     //Line 5
      b1.M1( );     //Line 6
    }
```

A) Compilation Error at Line 1.
B) Compilation Error at Line 2.
C) Compilation Error at Line 3.
D) Compilation Error at Line 4.
E) Compilation Error at Line 5.
F) Compilation Error at Line 6.
G) The code compiles successfully.

12) Which of the following most closely describes the process of overriding?

A) A class with the same name replaces the functionality of a class defined earlier in the hierarchy.
B) A function with the same name replaces the functionality of a function defined earlier in the inheritance hierarchy.
C) A function with the same name but different parameters gives multiple uses for the same function name.
D) Making a class abstract so that no objects can be declared from it.

13) "A plane is a machine that has a motor and has wings".
"A refrigerator is a machine that has a motor and has shelves".
Which of the following best describes the previous statements as a set of classes?

A) 1 class: A machine class that has an attribute for the type of machine.
B) 2 classes: A plane class that has two attributes, and a refrigerator class that also has two attributes.
C) 3 classes: A machine class that has one attribute: motor. A plane class that inherits from the machine class. And a refrigerator class that inherits from the plane class.
D) 3 classes: A machine class that has one attribute: motor. A plane class that inherits from the machine class. And a refrigerator class that also inherits from the machine class.

14) If we did not specify a constructor to the class, then :

A) we won't be able to create object of class
B) we won't be able to create object of class, and compiler will give compilation error
C) we won't be able to create object of class, and compiler will give warning
D) it will generate run-time error
E) None of the above

N *n;

15) Assume you have a class M that contains a pointer to an object of class N. Assume that we declare an object of M in the main( ) function. When will the body of the constructor of class N be executed?

A) When any member function of the class M is called.
B) After the body of the constructor of class M is executed.
C) Before the body of the constructor of class M is executed.
D) None of the above.

16) What will be the output when you compile and run the following piece of code?

```
class Parent
    {
     protected:
        int myVar;
     public:
        Parent(int x) { myVar=x; }
        void powerTwo(){ cout<<myVar*myVar; }
        virtual void powerThree() { cout <<myVar*myVar*myVar; }
    };
class Child:public Parent
    {
     protected:
        int myData;
```

```
        public:
            Child(int a, int b) : Parent(a) {myData= b;}
            void powerTwo() { cout <<myData*myData; }
            void powerThree() { cout<<myData*myData*myData; }
        };
    void main()
    {
      Child myCh(2,3);
      Parent *myPtr;
      myPtr = &myCh;
      myPtr->powerTwo(); //Line1
      myPtr->powerThree(); //Line 2
}
```

A) 4 8
B) 4 27
C) 9 27
D) 9 8
E) Compilation Error at Line1
F) Compilation Error at Line2

17) In order for the following piece of code to compile successfully, what are the constructors that are expected to exist in the Base class?

```
class Child : public Base
    {
        public:
            Child(int x)  { }
            Child(int x, int y) : Base(x,y) { }

    };
```

A) Base( ) and Base(int , int).
B) Base( ) and Base(int).
C) Base(int) and Base(int , int).
D) Base(int , int).

18)

```
class Stack
  {
    int tos,size;
    int * st;
    Stack( int s=5) { tos=0;size=s; st=new int[size];}
    ~ Stack( ){ delete []st;}
  };
void main()
{
```

```
}
```

what shall we add to the above class Stack to declare another object s2 from class Stack where s2 is declared in terms of s1; Stack s2(s1)?

A) We must specify overload of assignment operator for class Stack.
B) We must define a copy constructor to class Stack
C) A and B
D) This situation cannot be achieved in C++, however, it has been solved in other programming languages.

19) class Point

```
{
 float x, y;
 Point (float a, float b) { x=a; y=b;}
 Point () { x=0; y=0; }
}
```

To write copy constructor to class point, what would be its signature?

A) Point (Point)
B) Point (Point &)
C) Point & Point (Point &).
D) Point & Point (Point)
E) void Point (Point)
F) void Point (Point &)
G) None of the above

20) Assume you have a member function with the following prototype?

void myFunc(int x) ;

Which of the following are valid ways to overload it?

A) void myFunc(char ch) ;
B) int myFunc(int x) ;
C) void myFunc(char c1, char c2) ;
D) float myFunc(int x, int y) ;

21)What will be the output when you compile and run the following piece of code?

```
class Base
    {  public:
            Base() { cout<<"Welcome     " ; }
    };
```

```
class Derived : public Base
        { public:
                Derived() { cout<<"Hello     " ; }
        };
void main()
      {  Base myBase ;
         Derived myDerived ;
      }
```

A) Welcome Hello
B) Hello Welcome
C) Welcome Hello Welcome
D) Welcome Welcome Hello

22) What does the following piece of code do?

```
void main( )
        {
         float *ptr ;
        ptr = new float[15];
        }
```

A) Allocate space for a float variable that is not initialized
B) Allocate space for an array of 15 float elements that are not initialized
C) Allocate space for an array of 15 float elements that is initialized by the value 0
D) Allocate space for an array of 15 float elements where all the elements are initialized by the value 15
E) Compiler Error.

23) Which of the following statements are true about constructor?

A) A constructor can be overloaded.
B) A constructor is a special member function with the same name of the class.
C) A constructor can return a primitive or an object reference.
D) All the above

24) What will be the output when you compile and run the following piece of code?

```
class GrandFather
        { public:
                virtual void displayStuff() = 0 ;
                virtual void sayThings() = 0 ;
        };
class Parent : public GrandFather
```

```
        { public:
                void displayStuff( ) { cout<<"Parent's Stuff" ; }
        };
class Child : public Parent
        { public:
                void sayThings( ) { cout<<"Child's Things" ; }
        };
class GrandChild : public Child
        { public:
                void displayStuff( ) { cout<<"GrandChild's Stuff" ; }
                void sayThings( )    { cout<<"GrandChild's Things" ; }
        };
void main( )
        { GrandFather myGF ; // Line 1
          Parent myP ;       // Line 2
          Child myCh ;       // Line 3
          GrandChild myGC ;  // Line 4
          GrandFather * ptr ; // Line 5
        }
```

A) Compiler Error at Line 1.
B) Compiler Error at Line 2.
C) Compiler Error at Line 3.
D) Compiler Error at Line 4.
E) Compiler Error at Line 5.
F) The code compiles successfully.

25) What will be the output when you compile and run the following piece of code?

```
class Tester
 {  public:
            int x ;
            static int var ;
            Tester(int a) { x = a ; }
            static void myFunction(int a)
              {
                Tester obj(9) ;      // Line 1
                obj.x=a ;            // Line 2
                cout<<obj.x;
              }
 };
 int Tester::var=0;
void main()
{
 Tester myT(7) ;
 Tester::myFunction(15);
 cout<<myT.x ;
```

var=1 , x=7

```
cout<<Tester::var<<endl ;        // Line 3
}
```

A) 15 7 0
B) 7 15 0
C) 15 9 0
D) 9 15 0
E) Compilation error at line 1
F) Compilation error at line 2
G) Compilation error at line 3
H) None of the above.

Write all your answers for Question #1 and Question#2 in the table below.
Just write the letter(s) corresponding to your answer in the corresponding cell.
Answers that are not written in this table will NOT be marked nor considered.
Extra sheets of paper are NOT allowed to be attached to this exam.
You are NOT allowed to leave the room with the questions sheet.

Answer Sheet

Question #1 (True or False)	Question #2 (Choose all that apply)			
False	1.	C, F	15.	B
False	2.	A, B	16.	B
True	3.	B, C	17.	D
False	4.	B	18.	B
False	5.	A	19.	B
False	6.	B, C	20.	A, C
False	7.	C	21.	D
False	8.	C	22.	B
False	9.	D	23.	A, B
False	10.	A	24.	A, B
	11.	G	25.	E
	12.	B		
	13.	D		
	14.	B		

Q.1 True or False: (10 Marks)

1) The constructor of a class is responsible for removing the object from the memory (X)

2) Multilevel Inheritance is allowed as an object Oriented concept, but it is not allowed in C++. (X)

3) Association is a special type of Composition (X)

4) The term "Encapsulation" refers to an object of a class that contains another object of another class inside it. (X)

5) When inheriting from a Base class, the Derived class will inherit only the protected, and public members of the Base class. (✓)

6) Static member variable can only be modified through static member functions. (X)

7) If a certain function is made friend for class A, then that function can access only the private members of class A. (X)

8) Composition is special type of Inheritance; composition is stronger relation between 2 objects than inheritance. (X)

9) The "this" pointer is an array of pointers to all the objects created from a class. (X)

10) When overloading a certain function, it is obligatory that we specify a different number of parameters for the new function. (X)

Q.2- Choose the correct answer(s) (Choose all that apply) (50 Marks)

13:21

Object Oriented Programming using C++ (Menoufeya Branch) Time Allowed : 75 Mins

Choose (True or False)

3. The friend function is a Member function which is allowed to access the private members of a class. (1 Point)

- (●) True
- ( ) False

4. The copy constructor is called when a function return a reference to object(1 Point)

- (●) True
- ( ) False

5. It is not recommended to make the function inline when the function is recursive.(1 Point)

- ( ) True
- (●) False

6. The copy constructor is used to create a new object similar to an old one(1 Point)

01:13:17

7. The static members in a class can be used in static Methods of the same class only.(1 Point)

- ( ) True
- (●) False

8. the effect of private inheritance and protected inheritance appears on the child of first level of inheritance(1 Point)

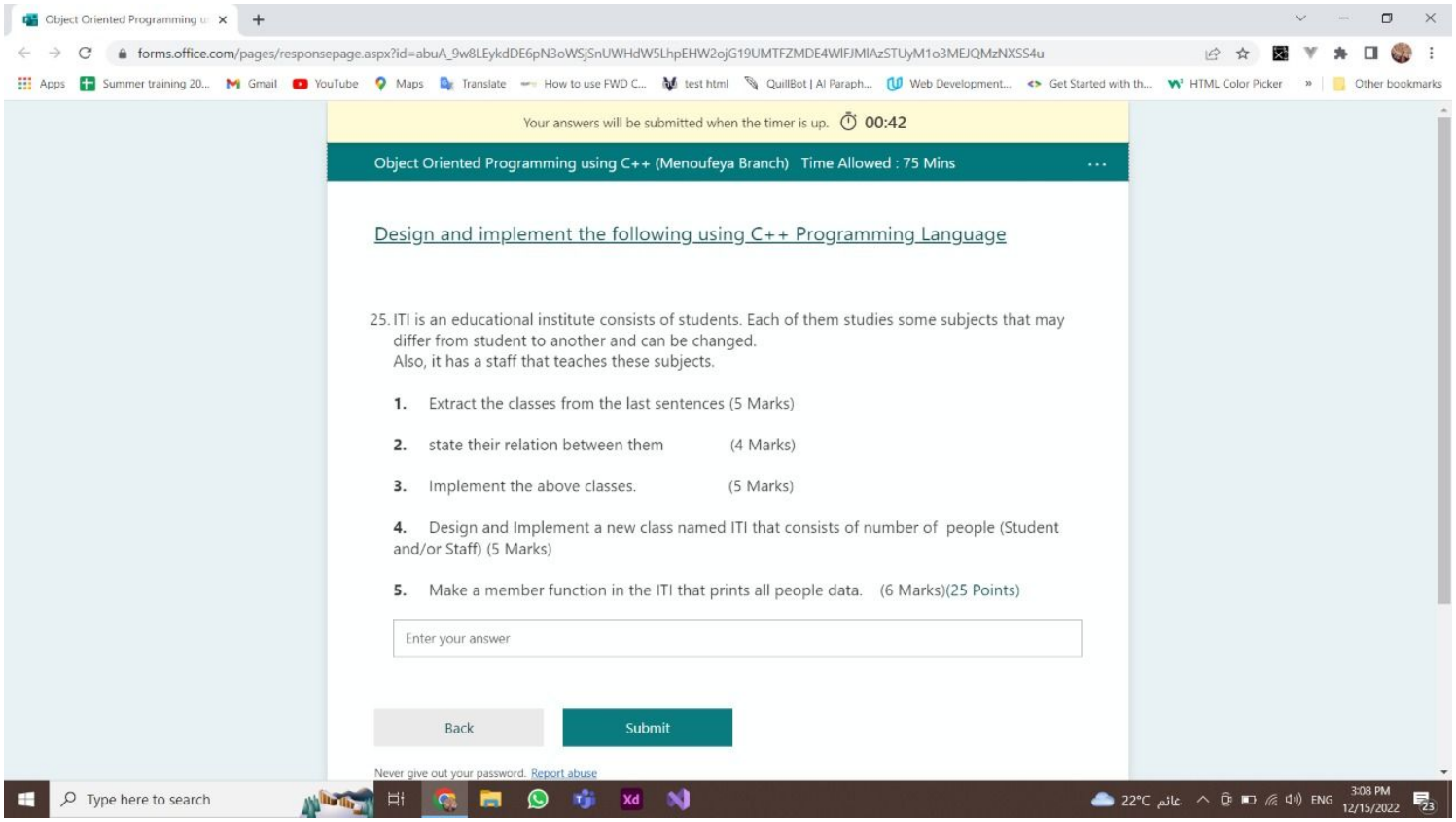
- ( ) True
- ( ) False

9. We use the inheritance for extending the base class and increasing more member functions and member variables(1 Point)

- ( ) True
- ( ) False

10. The operator overloading is one of the ways of polymorphism.(1 Point)

- ( ) True
- ( ) False



Your answers will be submitted when the timer is up. 00:42

Object Oriented Programming using C++ (Menoufeya Branch) Time Allowed : 75 Mins

Design and implement the following using C++ Programming Language

25. ITI is an educational institute consists of students. Each of them studies some subjects that may differ from student to another and can be changed.
Also, it has a staff that teaches these subjects.

1. Extract the classes from the last sentences (5 Marks)
2. state their relation between them (4 Marks)
3. Implement the above classes. (5 Marks)
4. Design and Implement a new class named ITI that consists of number of people (Student and/or Staff) (5 Marks)
5. Make a member function in the ITI that prints all people data. (6 Marks)(25 Points)

Enter your answer

Back Submit

Never give out your password. Report abuse

1- Extract the classes from the last sentences:
- Subject
- Person (Abstract)
- Student
- Staff

2- State their relation between them:
- Student and Staff are derived from Person (Inheritance).
- Student has a composition relationship with Subject
- Staff has a composition relationship with Subject
- ITI has an association relationship with Person

```
class Subject
{
  char name[20];
  public:
    Subject(){}
    Subject(char n[20])
    {
      strcpy(name,n);
    }
    char* GetName(){
      return name;
    }
    void SetName(char n[20])
    {
      strcpy(name,n);
    }
};

class Person
{
  protected:
    char name[20];
  public:
    Person(char n[20])
    {
      strcpy(name,n);
    }
    char* GetName(){
      return name;
    }
    void SetName(char n[20])
    {
      strcpy(name,n);
    }
    virtual void PrintInfo()=0;
};

class Student: public Person
{
  Subject subjects[7];
  int SubjectCount;
  public:
    Student(char* n):Person(n)
    {
      SubjectCount=0;
    }
    Student(char* n,char* subjectsNames[7], int number):Person(n)
    {
      if(number<=7){
        for(int i=0;i<number;i++)
        {
          subjects[i].SetName(subjectsNames[i]);
        }
        SubjectCount=number;
      }
      else{
        cout<<"Sorry, You tried add more than 7 subjects"<<endl;
        SubjectCount=0;
      }
    }
    void AddSubject(char* subjectName){
      if(SubjectCount<7){
        subjects[SubjectCount].SetName(subjectName);
        SubjectCount++;
      }
      else
      {
        cout<<"Sorry, the student can't have more than 7 subjects"<<endl;
      }
    }
    void SetSubject(char* subjectName,int index){
      if(SubjectCount>index)
      {
        subjects[index].SetName(subjectName);
      }
      else
      {
        cout<<"This Subject doesn't exist"<<endl;
      }
    }
    void PrintInfo()
    {
      cout<<"Name: "<<name<<endl;
      cout<<"Role: Student"<<endl;
      if(SubjectCount!=0){
        for(int i=0;i<SubjectCount;i++)
        {
          cout<<"Subject "<<i+1<<": ";
          cout<<subjects[i].GetName()<<endl;
        }
      }
      else
      {
        cout<<"The student isn't enrolled in any subject";
      }
      cout<<"#########################################"<<endl;
    }
};

class Staff: public Person
{
  Subject subjects[3];
  int taughtSubjectCount;
  public:
    Staff(char* n):Person(n)
    {
      taughtSubjectCount=0;
    }
    Staff(char* n,char* subjectsNames[3], int number):Person(n)
    {
      if(number<=3){
        for(int i=0;i<number;i++)
        {
          subjects[i].SetName(subjectsNames[i]);
        }
        taughtSubjectCount=number;
      }
      else{
        cout<<"Sorry, You tried add more than 3 subjects"<<endl;
        taughtSubjectCount=0;
      }
    }
    void AddSubject(char* subjectName){
      if(taughtSubjectCount<3){
        subjects[taughtSubjectCount].SetName(subjectName);
        taughtSubjectCount++;
      }
      else
      {
        cout<<"Sorry, the staff can't teach more than 3 subjects"<<endl;
      }
    }
    void SetSubject(char* subjectName,int index){
      if(taughtSubjectCount>index)
      {
        subjects[index].SetName(subjectName);
      }
      else
      {
        cout<<"This Subject doesn't exist"<<endl;
      }
    }
    void PrintInfo()
    {
      cout<<"Name: "<<name<<endl;
      cout<<"Role: Staff"<<endl;
      if(taughtSubjectCount!=0){
        for(int i=0;i<taughtSubjectCount;i++)
        {
          cout<<"Subject "<<i+1<<": ";
          cout<<subjects[i].GetName()<<endl;
        }
      }
      else
      {
        cout<<"The Staff isn't teaching any subject";
      }
      cout<<"#########################################"<<endl;
    }
};

class ITI
{
  Person** People;
  int count;
  public:
    ITI(Person** p,int size)
    {
      count=size;
      People= new Person*[count];
      for(int i=0;i<count;i++)
      {
        People[i]=p[i];
      }
    }
    void PrintAllPeopleInfo()
    {
      for(int i=0;i<count;i++)
      {
        People[i]->PrintInfo();
      }
    }
};


int main()
{
  char* subjects1[7]={"Math1","Math2","Math3","Math1","Math2"};
  char* subjects2[3]={"Mah1","Mah2","Mah3"};
  Student st1("Mahmoud",subjects1,5);
  Staff sf1("Ahmed",subjects2,3);
  Person* people[2]={&st1,&sf1};
  ITI obj(people,2);
  st1.AddSubject("Math8");
  st1.SetSubject("Mahmoud",7);
  st1.PrintInfo();
  sf1.PrintInfo();
  obj.PrintAllPeopleInfo();
  return 0;
}
```

3.What will be the output for the following code:

```
class Base
{
        protected:
                float x;
        public:
                Base(float m)
                {x = m;}
                virtual void PrintTotal()
                {cout<< "Total is: "<<x;}
};
class Derived : public Base
{
        protected:
                float y;
        public:
                Derived(float a, float b) : Base(a)
                {y = b;}
                void PrintTotal()
                {cout<<"Total is: "<<x + y;}
                void ShowValues()
                {cout<<"x= " <<x<<" and y = " <<y;}
};
void main()
{
        Derived myObj(7, 2);
        Base * ptr;
        ptr = &myObj;
        ptr->ShowValues();
}
```

(2 Points)

a. Total is: 2.

b. Total is: 7.

c. Total is: 9.

d. x = 7 and y = 2.

e. None of the above

4.Consider the following code:

```
class Base
{
        protected:
                float x, y;
        public:
                Base()
                {x = 2 ; y = 1 ; }
                Base(float a, float b)
                {x = a; y = b;}
                void SetX(float a) {x = a;}
                void SetY(float b) {y = b;}
                virtual void printSum()=0;
};
class Derived : public Base
{
        protected:
                float z;
        public:
                Derived(float a, float b, float c) : Base()
                {z = c;}
                void SetZ(float c) {z = c;}
                void showSum()
                {cout << x + y +z;}
};
void main()
{
        Base *myPtr;
        Derived myDerived(15, 5,4);
        myDerived.showSum();
}
```

The above code output is:

(2 Points)

a. 24

b. 7

c. Compilation Error

d. None of the above

5.Consider the following class:

```
class Test
{
    public:
            int x;
                Test(){x=0;}
                Test(int y)
            {
                x=y++;
            }
                Test(Test &r)
            {
                x=++r.x;
            }
            void print ()
            {
                cout<<x;
            }
};
void main()
{
    Test t(1);
    t.print();
    Test x(t);
    x.print();
    t.print();
}
```

What is the output of the previous program?
(2 Points)

a. 121
b. 122
c. 222
d. 233

6.Consider the following class:

```
class AA
{
  protected:
    int a;
  public:
    AA(int m) {a = m;}
};
class BB : public AA
{
  protected:
    int b;
  public:
    BB(int m, int n) : AA(n)
    {b = m;}
};
class CC : public AA
{
  protected:
    int c;
  public:
    CC(int m, int n) : AA(n)
    {c = m;}
};
class XX : private BB, protected CC
{
  public:
    XX(int m, int n, int p) : CC(n, p), BB(m,n)
    {}
};
        XX myObject(3, 4, 5);
```

The previous code what will be the values of a, b, c:
(2 Points)

a. a = 4, b = 3, c =4
b. a = 5, b = 3, c =4
c. Undetermined Values
d. Compliation Error

7.Consider the following class:

```
#include <iostream.h>
class polygon
{
   protected:
        int width, height;
   public:
        void set_values (int a, int b)
        {
                width = a; height = b;
        }
};
class output1
{
   public:
        void output (int i);
};
void output1::output (int i)
{
        cout << i <<"  ";
}
class rectangle: public polygon, public output1
{
   public:
        int area ()
        {
                return (width * height);
        }
};
class triangle: public polygon, public output1
{
   public:
        int area ()
        {
                return (width * height / 2);
        }
};
int main ()
{
        rectangle rect;
        triangle trgl;
        rect.set_values (4, 5);
        trgl.set_values (4, 5);
```

```
        rect.output (rect.area());
        trgl.output (trgl.area());
        return 0;
}
```

The previous code output:

(2 Points)

20

b) 10

c) 20 10

d) None of the mentioned

8.Consider the following class:

```
#include <iostream.h>
class Base
{
   public:
        virtual void print() = 0;
};
class DerivedOne : public Base

   public:
        void print()
        {
                cout << "DerivedOne\n";
        }
};
class DerivedTwo : public Base
{
   public:
        void print()
        {
                cout << "DerivedTwo\n";
        }
};
class Multiple : public DerivedOne, public DerivedTwo
{
   public:
        void print()
        {
                DerivedTwo :: print();
        }
};
int main()
{
        int i;
        Multiple both;
        DerivedOne one;
        DerivedTwo two;
        Base *array[ 3 ];
        array[ 0 ] = &both;
        array[ 1 ] = &one;
        array[ 2 ] = &two;
        array[ i ] -> print();
        return 0;
```

}
What is the output of the previous program?
(2 Points)
a) DerivedOne
b) DerivedTwo
c) Error
d) None of the mentioned

9.Consider the following class:

```
#include <iostream.h>
class student
{
   public:
        int rno , m1 , m2 ;
        void get()
        {
                rno = 15, m1 = 10, m2 = 10;
        }
};
class sports
{
   public:
        int sm;
        void getsm()
        {
                sm = 10;
        }
};
class statement:public student,public sports
{
        int tot,avg;
   public:
        void display()
        {
                tot = (m1 + m2 + sm);
                avg = tot / 3;
                cout << tot;
                cout << avg;
        }
};
int main()
{
        statement obj;
        obj.get();
        obj.getsm();
        obj.display();
}
```

What is the output of the previous program?
(2 Points)

a) 3100

b) 3010

c) 2010
d) 1010

10.Consider the following class:

```
#include <iostream.h>
class Base1
{
   protected:
        int SampleDataOne;
   public:
        Base1()
        {
                SampleDataOne = 100;
        }
        ~Base1() { }
        int SampleFunctOne()
        {
                return SampleDataOne;
        }
};
class Base2
{
   protected:
        int SampleDataTwo;
   public:
        Base2()
        {
                SampleDataTwo = 200;
        }
        ~Base2() { }
        int SampleFunctTwo()
        {
                return SampleDataTwo;
        }
};
class Derived1 : public Base1, public Base2
{
        int MyData;
   public:
        Derived1()
        {
                MyData = 300;
        }
        ~Derived1() { }
        int MyFunct()
```

```
        {
                return (MyData + SampleDataOne + SampleDataTwo);
        } };
int main()
{
        Base1 SampleObjOne;
        Base2 SampleObjTwo;
        Derived1 SampleObjThree;
        cout << SampleObjThree.Base1 :: SampleFunctOne() << endl;
        cout << SampleObjThree.Base2 :: SampleFunctTwo() << endl;
        return 0;
}
```

What is the output of the previous program?

(2 Points)

a) 100

b) 200

c) Both a & b

d) None of the mentioned

11.Consider the following class:

```
#include <iostream.h>
class stu
{
   protected:
        int rno;
   public:
        void get_no(int a)
        {
                rno = a;
        }
        void put_no(void)
        { }
};
class test:public stu
{
   protected:
        float part1,part2;
   public:
        void get_mark(float x, float y)
        {
                part1 = x;
                part2 = y;
        }
        void put_marks()
        { }
};
class sports
{
   protected:
        float score;
   public:
        void getscore(float s)
        {
                score = s;
        }
        void putscore(void)
        { }
};
class result: public test, public sports
{
        float total;
```

```
    public:
        void display(void);
};
void result::display(void)
{
        total = part1 + part2 + score;
        put_no();
        put_marks();
        putscore();
        cout << "Total Score=" << total << "\n";
}
int main()
{
        result stu;
        stu.get_no(123);
        stu.get_mark(27.5, 33.0);
        stu.getscore(6.0);
        stu.display();
        return 0;
}
```

What is the output of the previous program?
(2 Points)

a) 66.5
b) 64.5
c) 62.5
d) 60.5

12.Consider the following class:

```
#include <iostream.h>
class poly
{
   protected:
        int width, height;
   public:
        void set_values(int a, int b)
        {
                width = a; height = b;
        }
};
class Coutput
{
   public:
        void output(int i);
};
void Coutput::output(int i)
{
        cout << i;
}
class rect1:public poly, public Coutput
{
   public:
        int area()
        {
                return(width * height);
        }
};
class tri:public poly, public Coutput
{
   public:
        int area()
        {
                return(width * height / 2);
        }
};
int main()
{
        rect1 rect;
        tri trgl;
        rect.set_values(3, 4);
        trgl.set_values(4, 5);
```

```
        rect.output(rect.area());
        trgl.output(trgl.area());
        return 0;
}
```

What is the output of the previous program?

(2 Points)

a) 1212

b) 1210

c) 1010

d) none of the mentioned

13.Consider the following class:

```
#include <iostream.h>
class Person
{
protected:
	int id;
public:
	Person(int i)
	{
		id = i;
	}
	Person(Person &p)
	{
		id = p.id;
	}
	int GetId()
	{
		return id;
	}
	void SetId(int m)
	{
		id = m;
	}
};
class Employee: public Person
{
	int salary;
public:
	Employee(int i, int s) :Person(i)
	{
		salary = s;
	}
	Employee(Employee &e)
	{
		salary = e.salary;
	}
	int GetSalary()
	{
		return salary;
	}
	void SetSalary(int m)
	{
		salary = m;
```

```
        }
};
Employee FillEmp()
{
        Employee em(3, 4);
        return em;
}
int main()
{
        Employee emp(7, 8);
        emp = FillEmp();
        cout << emp.GetId() ;
        cout << emp.GetSalary() << endl;
        return 0;
}
```

What is the output of the previous program?
(2 Points)

a. 34
b. 78
c. Compile error
d. Non of the above

14.

```
#include <iostream.h>
class A
{
	public:
		A(int n = 0)
		{
			m_n = n;
		}
		A(A& a)
		{
			m_n = a.m_n;
			++m_copy_ctor_calls;
		}
		static int m_copy_ctor_calls;
	private:
		int m_n;
};

int A::m_copy_ctor_calls = 0;
A f(A& a)
{
	return a;
}
A g(A a)
{
	return a;
}

int main()
{
	A a;
	A b = a, c(a);
	cout << A::m_copy_ctor_calls; //2
	b = g(c);
	cout << A::m_copy_ctor_calls; //4
	A& d = f(c);
	cout << A::m_copy_ctor_calls <<endl; //5
	return 0;
}
```

What is the output of the above code?
(2 Points)
a) 244

b) 245
c) 254
d) 255

```
15.#include <iostream.h>
class Person
{
public:
	Person()
	{
		cout << "Default constructor of Person"<<endl;
	}
	Person(Person& p)
	{
		cout << "Copy constructor of Person" << endl;
	}
	~Person()
	{
		cout << "Destructor of Person" << endl;
	}
	Person operator = (Person &per)
	{
		cout << "Overloading of Person Equal Operator"<<endl;
		return *this;
	}
};
class Employee : public Person
{
public:
	Employee ()
	{
		cout << "Default Constructor of Employee" << endl;
	}
	Employee(Employee& e)
	{
		cout << "Copy Constructor of Employee" << endl;
	}
	~Employee()
	{
		cout << "Destructor of Employee" << endl;
	}
	Employee operator = (Employee e)
	{
		cout << "Overloading of Employee Equal Operator"<<endl;
		return (*this);
	}
};
```

```
Employee FillEmp()
{
        Employee emp;
        cout << "This call of Fill Employee" << endl;
        return emp;
}
void PrintEmp(Employee e)
{
        cout << "This call of Print Employee" << endl;
}
int main()
{
        Employee Emp;
        Emp = FillEmp();
        PrintEmp(Emp);
        return 0;
}
```

what is the number of printed statement when executing the previous code

(2 Points)

a) 24

b) 26

c) 23

d) 20

e) 22

16.When calling the overload of the = operator of a child, it call the overload = operator of the parent:
(2 Points)
a) Before the =operator of the child
b) After the =operator of the child
c) Don't call the overload of the parent
d) None of the above

17.When calling the copy constructor of a child, it call the copy constructor of the parent:
(2 Points)
a) Before the copy constructor of the child
b) After the copy constructor of the child
c) Don't call the constructor of the parent
d) None of the above

c if "constructor" means "copy constructor". it will be d otherwise.

18.When destructing a child object, the destructor of the parent is called:
(2 Points)
a) Before the destructor of the child
b) After the destructor of the child
c) Don't call the destructor of the parent
d) None of the above

19.

```
class Employee
{
	public:
		Employee()
		{
			cout<<"Default Constructor"<<endl;
		}
~Employee()
		{
			cout<<"Destructor"<<endl;
		}
Employee(Employee& rEmp)
		{
			cout<<"Copy Constructor"<<endl;
		}
		Employee operator = (Employee e)
		{
			cout<<"Overloading = Operator"<<endl;
			return e;
		}
};
Employee FillEmp()
{
	Employee e;
	cout<<"This call of Fill Employee"<<endl;
	return e;
}
void PrintEmp(Employee e)
{
	cout<<"This call of Print Employee"<<endl;
}
int main()
{
	Employee Emp;
	Emp = FillEmp();
	PrintEmp(Emp);
	return 0;
}
```

what is the number of function call (member and non member) when executing the previous code

(2 Points)

a) 10

b) 11

c) 12
d) 13

20.To use an object from ofstream to write the data of a user define class on a file, we can:
(2 Points)
a) Overload the insertion operator as a non member method for the used define class
b) Overload the insertion operator as a member in the ofstream class
c) Override the insertion operator as a non member method for the used define class
d) Both a and
e) None of the above

C++ Final Exam

Duration: 0.5 hr.

Name : Group :

Choose only one answer.

1. void main()

```
{
    int friend;
    friend+=5;
}
```

The above code is

a) C valid only.

b) C and C++ valid.

c) C++ valid only.

d) C and C++ invalid

2. int a;

```
void main()
{
int a;
a=50;
::a=20;
cout << a;
}
```

The above piece of code will print:

a) The value of 20.

b) Compiler will give an error.

c) The Value of 50.

d) None of above.

3. The following function prototype uses the default argument

   void myfunc(int x=3,int y);

   Is the above function valid?

   a) Yes, you can give a default parameter for x and not for y.
   b) No, both x and y must have a default values.
   c) No, you can give a default parameter for y and not for x.
   d) No, there must be no arguments without defaults to the right of the default arguments.

4. void main()

```
{
    int *p;
    p=new int(5);
}
```

   The above code

   a) Allocates an integer that is not initialized.
   b) Allocates an array of integers of 5 elements that is not initialized.
   c) Allocates an integer that is initialized by 5.
   d) Allocates an array of integers where all its elements is initialized by 5.

5. class myclass

```
{
    int x;
protected:
int y;
};
void main()
{
    myclass c1;
    c1.x=100;
    c1.y=500;
}
```

   The above code will produce

   a) An error, x is private.

b) An error, y is protected.

c) No error

d) Both (a) and (b)

6. The class constructor must

a) Return void.

b) Have the same name as class.

c) Have no parameter.

d) Both (a) and (b).

7. Which type of stand alone functions is allowed to access the private members of a class?.

a) Template Function.

b) Inline Function.

c) Friend Function.

d) Overloaded Function.

8. class base

```
{
    int x;
protected:
    int y;
public:
    int c;
};
class derived:private base
{
//some code
};
```

Class derived can access:

a) Private and protected members of base class.

b) Only private members of base class.

c) Only protected members of base class.

d) Protected and public members of the base class.

9. class base

```
{
public:
    base(){cout << "Constructing base class\n";}
};
class derived:public base
{
public:
    derived(){cout << "Constructing derived class\n";}
};
void main()
{
derived d;



}
```

The following code will print:

a) Constructing base class
b) Constructing base class
   Constructing derived class
c) Constructing derived class
d) Constructing derived class
   Constructing base class

10. class base

```
{
public:
    void say() {cout << "\n I'm base \n";}
};
class derived:public base
{
public:
    void say() {cout << "\n I'm derived \n";}
```

```
};
void main()
{
base *pb;
base mybase;
derived myderived;

pb=&mybase;
pb->say();
pb=&myderived;
pb->say();
}
```

The previous code will print:

a) I'm base
   I'm derived
b) I'm base
   I'm base
c) I'm derived
   I'm derived
d) Non of the above

11. Passing reference to object as a function parameter in order to
    a) The destructor not to be called
    b) The constructor not to be called
    c) Make a call by value function call
    d) Have no effect
12. Operator Overloading
    a) Changes the precedence of the operator
    b) Extends the functionality of the operator
    c) Changes the default function of the operator
    d) Has no effect on the operator

13. The protected members for the class user is equivalent to
    a) Public Member
    b) Private Member
    c) Global Variable
    d) Local Variable

14. The constructor is called with
    a) Declaring a class
    b) Defining an object of the class
    c) Calling the constructor function
    d) Removing the object from the program memory

15. The class attribute can be charcterized by:
    a) Every object has it's own value of the class attribute
    b) All objects of the class have the same value
    c) Make the variable scope within the class
    d) Illegal definition

16. The virtual base class
    a) Can be accessed through a pointer to base class
    b) Can not create an object of it.
    c) Is used to avoid multiple instance in multiple inheritance
    d) Which contains pure virtual functions

17. Structure Objects are
    a) Objects holding information of system
    b) Objects deals with file data structures
    c) Objects support communication with users
    d) All of the above

18. Aggregation is
    a) A special type of inheritance
    b) A special type of association

c) Is similar to association

d) Is similar to inheritance

20. A one way association with multiplicity many is represented by:

a) One pointer in the one side

b) Set of pointers in the one side

c) One pointer in the either sides.

d) Set of pointer in the either sides.

OOP exam - Assiut PWD ...

True or False (10 marks)

2

The destructor of a class is responsible for removing the object from the memory
(1 Point)

- True
- False

3

Multilevel Inheritance is allowed as an object Oriented concept, but it is not allowed in C++
(1 Point)

- True
- False

4

Association is a special type of Composition
(1 Point)

- True
- False

5

The term "Encapsulation" refers to an object of a class that contains another object of another class inside it
(1 Point)

- True
- False

6

When inheriting from a Base class, the Derived class will inherit only the protected, and public members of the Base class
(1 Point)

- True
- False

7

Static member variable can only be modified through static member functions
(1 Point)

- True
- False

8

If a certain function is made friend for class A, then that function can access only the private members of class A
(1 Point)

- True
- False

9

Composition is special type of Inheritance; composition is stronger relation between 2 objects than inheritance
(1 Point)

- True
- False

10

The "this" pointer is an array of pointers to all the objects created from a class
(1 Point)

- True
- False

11

When overloading a certain function, it is obligatory that we specify a different number of parameters for the new function
(1 Point)

- True
- False

Back | Next | Page 2 of 3

Never give out your password. Report abuse

This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.

Powered by Microsoft Forms | Privacy and cookies | Terms of use

Choose the correct answer(s) : (50 Marks)

12

Which of the following is true about an object member function?
(2 Points)

- A) It can be called using the name of the class
- B) It can access static variables of the class
- C) It has a "this" pointer as an implicit parameter passed to it
- D) It can access the instance variables
- E) It cannot be overloaded
- F) It can call other member functions from inside it
- A & B & C
- B & C & E
- A & C & D & F
- B & C & D & F

13

Which of the following is true about the function prototype below?

void myFunc (int myDef=17, int myVar , int myNormalVar=5) ;
(2 Points)

- We should also give a default value to myVar
- We must only give a default parameter for myNormalVar and not the others
- The function is correct in that way

14

Which steps will allow Sub to compile?
(2 Points)

```
class Super
 {
    protected:
            Super (int a)
               {
                 this.a = a;     // Line 1
               }
   private:                      // Line 2
            int a;
 };
 class Sub: public  Super
 {
     public:
            Sub (int a) : Super (a)  { }
     public:
            Sub()              //Line 3
            {
              this.a= 5;       //Line 4
            }
 };
```

- A) Class Sub compile successfully
- B) Comment Line 2
- C) In Line 1 and 4 to, use (*this).a instead of this.a;
- D) Change Line 1 and 4 to, this(a);
- E) Change Line 3 to, Sub(): Super(5)
- F) Change Line 3 to, Sub(): this(5)

B & D

B & C & E

B & D & F

15

```
class Test
{
    int x;
public:
    Test( ) {  x = 0;   }
    Test(int y) {  x = y++; }
    Test(Test &r)   {  x = ++r.x;  }

    void print( )
    {
        cout<<x ;
    }
};
void main()
{
    Test t(1) ;
    t.print() ;
    Test x(t) ;
    x.print() ;
    t.print() ;
}
```

What will be the output when you compile and run the following piece of code?
(2 Points)

121

122

222

223

16

What will happen here
(2 Points)

```
class Parent
{
public:
    int x;
    Parent(int m)   { x = m ; }
};

class Child : protected Parent
{
public:
    int y;
    Child(int m, int n) : Parent(m) {   y = n ;  }
};

class GrandChild : public Child
{
    int z ;
public:
    GrandChild(int a, int b, int c) : Child(a,b) {  z = c ; }
};

void main( )
{
        GrandChild obj(3,5,7);
        cout<<"Value of x is: "<<obj.x <<endl ;          //Line 1
        cout<<"Value of y is: "<<obj.y <<endl ;          //Line 2
        cout<<"Value of z is: "<<obj.z <<endl ;          //Line 3
}
```

A) Compiler Error at Line 1

B) Compiler Error at Line 2

B) Compiler Error at Line 2

C) Compiler Error at Line 3

D) The code compiles successfully

A & C

B & C

A & B

17

What will be the output when you compile and run the following piece of code?
(2 Points)

```
class Parent
{
    int x;
    Parent(int m){ x = m ; }
};

class Child : public Parent
{
    public:
        int y;
        Child(int m, int n) : Parent(m) //Line 1
        {
            y = n ;
        }
};

void main( )
{
    Child obj(3,5,7);            //Line 2
    cout<<"Value of x is: "<<obj.x <<endl ;  //Line 3
    cout<<"Value of y is: "<<obj.y <<endl ;  //Line 4
}
```

A) Compiler Error at Line 1

B) Compiler Error at Line 2

C) Compiler Error at Line 3

D) The code compiles successfully

A & B

A & B & C

B & C

18

In order to turn a class into an abstract class, which of the following do we need to do?
(2 Points)

A) Write the abstract keyword before the name of the class

B Make the class a pure virtual class

C) Write one or more pure virtual functions inside the class

D) A & C

E) None of the above

19

```
class Parent
{
    int y;
    static int z;
public:
    Parent( )
    {
        z=0;    // Line1
    }

    Parent (int a=5)    //Line 2
    {
        y=a;
    }
};

void main( )
{
    Parent d(4);      //Line 3
    Parent m;         //Line 4
}
```

What will be the output when you compile and run the following piece of code?
(2 Points)

- Compilation Error at Line 1, an object member function cannot access a static member
- Compilation Error at Line 2, constructor should initialize static member (z=0;)
- Compilation Error at Line 3
- Compilation Error at Line 4
- The code compiles successfully

20

What will be the output when you compile and run the following piece of code?
(2 Points)

```
class Parent
{
protected:
    int x;
public:
    Parent(int m)
    {
        x = m ;
    }

    friend void display( );
};

class Child : public Parent
{
private:
    int y;
public:
    Child(int m, int n) : Parent(m)
    {
        y = n ;
    }
};
```

```
void display ()
{
    Child c(3,4);
    cout <<"x="<<c.x<<"y="<<c.y; // Line 1
}

void main ()
{
    display();
}
```

- A) Compilation Error at Line 1, Child::x is inaccessible
- B) Compilation Error at Line 1, Child::y is inaccessible
- C) A and B
- D) The code compiles successfully

21

What will be the output when you compile and run the following piece of code?
(2 Points)

```
class Nice
{
    int a ;
public:
    Nice( ) { a = 0 ; }
    Nice(Nice & myN)
    {
```

```
        this->a =  myN.a ;
        cout<<"I am the copy constructor" ;
    }

    void setA(int m)
    {
        a = m;
    }

    int getA() { return a ; }
};

void show(Nice &obj)
{
        cout<<"I am the show function, value is: " << obj.getA() ;
}

void main()
{
    Nice n1;
    n1.setA(15) ;
    show(n1) ;
}
```

- I am the show function, value is: 15
- I am the show function, value is: 15 I am the copy constructor
- I am the copy constructor I am the show function, value is: 15
- I am the copy constructor

22

What will be the output when you compile and run the following piece of code?
(2 Points)

```
class A
{
     int x;
protected :
    int y ;
public :
    A(int x1=5,int y1=3)   { x=x1;     y=y1; }

    void M1( ) { cout <<"\n This is M1() in class A:Base class"; }

    void M3( ) { cout <<"\n This is M3() in class A:Base class"; }
};
class B : private A
{
    int w;
protected:
    int v;
public:
    B(int v1=3, int w1=9) { v=v1;   w=w1; }
    void M3()
    {
        M1( );   //Line 1
        y++;      //Line2
    }
};
```

```
class C: public B
{
public:
    void  M4( )
    {
        M1( );     //Line 3
        y++;       //Line 4
    }
};
void main()
{
    B b1;
    b1.M3( );        //Line 5
    b1.M1( );       //Line 6
}
```

- A) Compilation Error at Line 1
- B) Compilation Error at Line 2
- C) Compilation Error at Line 3
- D) Compilation Error at Line 4
- E) Compilation Error at Line 5
- F) Compilation Error at Line 6
- G) The code compiles successfully
- A & B & C
- C & D & F
- A & C & F

23

Which of the following most closely describes the process of overriding?

(2 Points)

- A class with the same name replaces the functionality of a class defined earlier in the hierarchy
- A function with the same name replaces the functionality of a function defined earlier in the inheritance hierarchy
- A function with the same name but different parameters gives multiple uses for the same function name
- Making a class abstract so that no objects can be declared from it

24

"A plane is a machine that has a motor and has wings".

"A refrigerator is a machine that has a motor and has shelves".

Which of the following best describes the previous statements as a set of classes?
(2 Points)

- 1 class: A machine class that has an attribute for the type of machine
- 2 classes: A plane class that has two attributes, and a refrigerator class that also has two attributes
- 3 classes: A machine class that has one attribute: motor. A plane class that inherits from the machine class. And a refrigerator class that inherits from the plane class
- 3 classes: A machine class that has one attribute: motor. A plane class that inherits from the machine class. And a refrigerator class that also inherits from the machine class

25

If we did not specify a constructor to the class, then :
(2 Points)

- we won't be able to create object of class
- we won't be able to create object of class, and compiler will give compilation error
- we won't be able to create object of class, and compiler will give warning
- it will generate run-time error
- None of the above

26

Assume you have a class M that contains a pointer to an object of class N. Assume that we declare an object of M in the main( ) function. When will the body of the constructor of class N be executed?
(2 Points)

- When any member function of the class M is called
- After the body of the constructor of class M is executed
- Before the body of the constructor of class M is executed
- None of the above

27

What will be the output when you compile and run the following piece of code?
(2 Points)

```
class Parent
{
protected:
    int myVar;
public:
    Parent(int x)
    {
        myVar=x;
    }
```

```
class Child : public Parent
{
protected:
    int myData;
public:
    Child(int a, int b) : Parent(a)
    {
        myData= b;
    }

    void powerTwo()
    {
        cout <<myData*myData;
    }
```

```
    void powerTwo()
    {
        cout<<myVar*myVar;
    }

    virtual void powerThree()
    {
        cout <<myVar*myVar*myVar;
    }
};
```

```
    void powerThree()
    {
        cout<<myData*myData*myData;
    }
};

void main()
{
    Child myCh(2,3);
    Parent *myPtr;
    myPtr = &myCh;
    myPtr->powerTwo(); //Line1
    myPtr->powerThree(); //Line 2
}
```

- 4 8
- 4 27
- 9 27
- 9 8
- Compilation Error at Line 1
- Compilation Error at Line 2

28

```
s Child : public Base

ic:
Child(int x)
{
}

Child(int x, int y) : Base(
{
}
```

In order for the following piece of code to compile successfully, what are the constructors that are expected to exist in the Base class?
(2 Points)

- Base( ) and Base(int , int)
- Base( ) and Base(int)
- Base(int) and Base(int , int)
- Base(int , int)

29

```
class Stack
{
    int tos,size;
    int * st;
    Stack( int s=5)
    {
        tos=0;size=s;
        st=new int[size];
    }

    ~ Stack( ){ delete []st;}
};
```

what shall we add to class Stack to declare another object s2 from class Stack where s2 is declared in terms of s1 => Stack s2(s1)?
(2 Points)

- A) We must specify overload of assignment operator for class Stack
- B) We must define a copy constructor to class Stack
- C) A and B
- D) This situation cannot be achieved in C++, however, it has been solved in other programming languages

30

```
class Point
{
    float x, y;
    Point (float a, float b)
```

```
    {
        x=a;
        y=b;
    }

    Point()
    {
        x=0;
        y=0;
    }
};
```

To write copy constructor to class Point, what would be its signature?
(2 Points)

- Point (Point)
- Point (Point &)
- Point & Point (Point &)
- Point & Point (Point)
- None of the above

31

Assume you have a member function with the following prototype?

void myFunc(int x) ;

Which of the following are valid ways to overload it?
(2 Points)

- A) void myFunc(char ch) ;
- B) int myFunc(int x) ;
- C) void myFunc(char c1, char c2) ;
- D) float myFunc(int x, int y) ;
- A & C
- A & C & D
- A & B & D

32

```
class Base
{
public:
    Base()
    {
        cout<<"Welcome  " ;
    }
};

class Derived : public Base
{
public:
    Derived()
    {
        cout<<"Hello    " ;
    }
};

void main()
{
    Base myBase ;
    Derived myDerived ;
}
```

What will be the output when you compile and run the following piece of code?
(2 Points)

- Welcome Hello
- Hello Welcome
- Welcome Hello Welcome

- Welcome Welcome Hello

33

What does the following piece of code do?

```
void main( )
{
    float *ptr ;
    ptr = new float[15];
}
```

(2 Points)

- Allocate space for a float variable that is not initialized
- Allocate space for an array of 15 float elements that are not initialized
- Allocate space for an array of 15 float elements that is initialized by the value 0
- Allocate space for an array of 15 float elements where all the elements are initialized by the value 15
- Compiler Error.

34

Which of the following statements are true about constructor?
(2 Points)

- A) A constructor can be overloaded.
- B) A constructor is a special member function with the same name of the class.
- C) A constructor can return a primitive or an object reference.
- D) All the above
- E) A & B
- F) A & C

35

What will be the output when you compile and run the following piece of code?
(2 Points)

```
class GrandFather
{
public:
    virtual void displayStuff() = 0 ;

    virtual void sayThings() = 0 ;
};


class Parent : public GrandFather
{
public:
    void displayStuff()
    {
        cout<<"Parent's Stuff" ;
    }
};
```

```
class Child : public Parent
{
public:
    void sayThings()    {  cout<<"Child's Things" ; }
};

class GrandChild : public Child
{
public:
    void displayStuff( )
    {
        cout<<"GrandChild's Stuff" ;
    }

    void sayThings( )
    {
        cout<<"GrandChild's Things" ;
    }
};

void main( )
{
    GrandFather myGF ;  // Line 1
    Parent myP ;                //  Line 2
    Child myCh ;                //  Line 3
    GrandChild myGC ;  //  Line 4
    GrandFather * ptr ;  //   Line 5
}
```

- A) Compiler Error at Line 1
- B) Compiler Error at Line 2
- C) Compiler Error at Line 3
- D) Compiler Error at Line 4
- E) Compiler Error at Line 5
- F) The code compiles successfully
- G) A & B

H) C & E

I) A & B & D

36

```
class Tester
{
public:
    int x ;
    static int var ;
    Tester(int a)
    {
        x = a ;
    }

    static void myFunction(int a)
    {
        Tester obj(9) ;          // Line 1
        obj.x=a ;                // Line 2
        cout<<obj.x;
    }
};
int Tester::var=0;

void main()
{
    Tester myT(7) ;
    Tester::myFunction(15);
    cout<<myT.x ;
    cout<<Tester::var<<endl ;          // Line 3
}
```

What will be the output when you compile and run the following piece of code?
(2 Points)

15 7 0

7 15 0

15 9 0

9 15 0

Compilation error at line 1

Compilation error at line 2

Compilation error at line 3

None of the above

Back | Submit | Page 3 of 3

Never give out your password. Report abuse

This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.

Powered by Microsoft Forms | Privacy and cookies | Terms of use

C++ Exam - Mobile Native (Alexandria 42) - Time Allowed: 1:45 Hours
Read Carefully the Question It is NOT Allowed to use any digital media It is NOT Allowed to use the internet
Required
1.Name in English:

2.How structures and classes in C++ differ?
(2 Points)

- a) In Structures, members are public by default whereas, in Classes, they are private by default
- b) In Structures, members are private by default whereas, in Classes, they are public by default
- c) Structures by default hide every member whereas classes do not
- d) Structures cannot have private members whereas classes can have

3.What will be the output of the following C++ code?

```
#include<iostream.h>
 class X
{
int m;
 public:
X() : m(10) { }
X(int mm): m(mm) { }
int getm() { return m; }
};
class Y : public X
{ int n;
public: Y(int nn) : n(nn) {}
int getn() { return n; } };
int main()
{
Y yobj( 100 );
 cout << yobj.getm() << " " << yobj.getn() << endl;
}
```

(2 Points)

- a) 10 100
- b) 100 10
- c) 10 10
- d) 100 100

4.What will be the output of the following C++ code?

```
 #include <iostream.h>
class BaseClass
{
protected: int i;
 public: BaseClass(int x) { i = x; }
 ~BaseClass() { }
```

```
};
class
DerivedClass: public BaseClass
{ int j;
public: DerivedClass(int x, int y): BaseClass(y) { j = x; }
 ~DerivedClass() { }
void show() { cout << i << " " << j << endl; }
 };
int main() {
DerivedClass ob(3, 4);
ob.show();
return 0; }
```

(2 Points)

- a) 3 4
- b) 4 3
- c) 4
- d) 3

5.What will be the output of the following C++ code?

```
 #include <iostream.h>
class MyInterface {
public:
virtual void Display() = 0;
};
 class Class1 : public MyInterface {
 public:
void Display() { int a = 5; cout << a; }
};
class Class2 : public MyInterface {
 public:
void Display() { cout <<" 5" << endl; } };
 int main() {
Class1 obj1;
 obj1.Display();
Class2 obj2;
obj2.Display();
return 0; }
```

(2 Points)

- a) 5
- b) 10
- c) 5 5
- d) 15

6.What will be the output of the following C++ code?

```
#include
class p
```

```
{
protected:
int width, height;
public:
void set_values (int a, int b)
{
width = a; height = b;
}
virtual int area (void) = 0;
};
class r: public p
{
public:
int area (void)
{
return (width * height);
}
};
class t: public p
{
public:
int area (void)
{
return (width * height / 2);
}
};
int main ()
{
r rect;
t trgl;
p * ppoly1 = &rect;
p * ppoly2 = &trgl;
ppoly1->set_values (4, 5);
ppoly2->set_values (4, 5);
cout << ppoly1 -> area() ;
cout << ppoly2 -> area();
return 0;
}
```

(2 Points)

- a) 1020
- b) 20
- c) 10
- d) 2010

7.What will be the output of the following C++ code?

```
#include <iostream.h>
#include <string.h>
 class A {
 float d;
 public:
virtual void func(){
cout<<"Hello this is class A\n"; }
};
class B: public A {
int a = 15;
public: void func(){
 cout<<"Hello this is class B\n"; }
};
 int main() {
A *a;
a->func();
return 0; }
```

(2 Points)

- a) Hello this is class A
- b) Hello this is class B
- c) Error
- d) Unexpected Execution

8.Virtual functions in C++ tells the compiler to perform _____ on such functions.

(2 Points)

- a) static binding
- b) late binding
- c) compile time binding
- d) no binding

9.Which of the following is correct?

(2 Points)

- a) Base class pointer object cannot point to a derived class object
- b) Derived class pointer object cannot point to a base class object
- c) A derived class cannot have pointer objects
- d) A base class cannot have pointer objects

10.Why references are different from pointers?

(2 Points)

- a) A reference cannot be made null
- b) A reference cannot be changed once initialized
- c) No extra operator is needed for dereferencing of a reference
- d) All of the mentioned

11.How many types of polymorphism are there in C++?

(2 Points)

- a) 1
- b) 2 [Static binding & Dynamic binding] or [early binding & late binding]
- c) 3
- d) 4

12.What is the difference between delete and delete[] in C++?

(2 Points)

- a) delete is used to delete normal objects whereas delete[] is used to pointer objects
- b) delete is a keyword whereas delete[] is an identifier
- c) delete is used to delete single object whereas delete[] is used to multiple(array/pointer of) objects
- d) delete is syntactically correct but delete[] is wrong and hence will give an error if used in any case

13.What will be the output of the following C++ code?

```
#include <iostream.h>
class A {
private: int a;
A() { a = 5;}
};
int main() {
A *obj = new A;
 cout << obj->a;
}
```

(2 Points)

- a) 5
- b) Garbage value
- c) Compile-time error
- d) Run-time error

14.What is dynamic binding?

(2 Points) https://www.geeksforgeeks.org/dynamic-binding-in-cpp/

- a) The process of linking the actual code with a procedural call during run-time
- b) The process of linking the actual code with a procedural call during compile-time
- c) The process of linking the actual code with a procedural call at any-time
- d) All of the mentioned

15.Which of the following is correct about this pointer in C++?

(2 Points) https://www.geeksforgeeks.org/this-pointer-in-c/

- a) this pointer is passed as a hidden argument in all the functions of a class
- b) this pointer is passed as a hidden argument in all non-static functions of a class
- c) this pointer is passed as a hidden argument in all static functions of a class
- d) this pointer is passed as a hidden argument in all static variables of a class

16.What will be the output of the following C++ code?

```
#include <iostream.h>
```

```
int main() {
int a = 5, b = 10, c = 15;
 int *arr[ ] = {&a, &b, &c};
cout << arr[1];
 return 0; }
```

(2 Points)

(*) + [ ] --> pointer to the array as a hole
[ ] --> pointer to the first element in the array

Remember C course ;)

- a) 5
- b) 10
- c) 15
- d) it will return some random number

17.What will be the output of the following C++ code?

```
#include <iostream.h>
int main() {
char *ptr; char Str[] = "abcdefg";
ptr = Str;
 ptr += 5;
 cout << ptr;
return 0; }
```

(2 Points)

- a) fg
- b) cdef
- c) defg
- d) abcd

18.

```
int result(int n)
{
   if (n == 1)
      return 2;
   else
      return 2 * result(n - 1); }
```

What value does result(5) return?

(2 Points)

- a) 64
- b) 32
- c) 16
- d) 8
- e) 2

```
result(5) =
2 * result(4)    (2*16) = 32
        |
2 * result(3)   (2 *8) = 16
        |
2 * result(2)  (2*4) = 8
        |
2 * result(1)  (2*2)  = 4
```

19.Consider the following class:

```
class Test
{
   public:
      int x;
      Test(){x=0;}
      Test(int y)
      {
```

```
        x=y++;    x = y ;
    }
    Test(Test &r)
    {
        x=++r.x;  x = r.x+1 ;
    }             and r.x is incremented
    void print ()
    {
        cout<<x;
    }
};

void main()
{
    Test t(1);
    t.print();    t.x = 1
    Test x(t);    copy constructor
    x.print();    x.x = 2  and t.x = 2
    t.print();
}
```

What is the output of the previous program?
(2 Points)

- 121
- 122
- 222
- 233

20.What will be the output of the following C++ code?

```
#include <iostream.h>
int main() {
int a[2][4] = {3, 6, 9, 12, 15, 18, 21, 24};
 cout << *(a[1] + 2) << *(*(a + 1) + 2) << 2[1[a]];
return 0; }  a[1][2]          a[1][2]          a[1][2]
```

(2 Points)

- a) 15 18 21
- b) 21 21 21
- c) 24 24 24
- d) Compile time error

21.What will be the output of the following C++ code?

```
 #include <iostream.h>
int main () {
 int numbers[5];
 int * p; p = numbers;
*p = 10;      value of first element in arr = 10
p++;          number[0] = 10
*p = 20;      number [1] = 20
```

```
p = &numbers[2];
*p = 30;                number [2] = 30
p = numbers + 3;
 *p = 40;               number [3] = 40
 p = numbers;
*(p + 4) = 50;          number [4] = 50
for (int n = 0; n < 5; n++)
cout << numbers[n] << ",";
return 0;
 }
```

(2 Points)

- a) 10,20,30,40,50,
- b) 1020304050
- c) compile error
- d) runtime error

22.What will be the Output of the following Code:

```
int * GetMin(int ar[5]);
int main ()
{
   int *ptr;
   int X[5] = {12, 18, -11, -20, 3};
   ptr = GetMin(X);
   printf("%d", *ptr);
   return 0;
}
int * GetMin(int ar[5])
{
   int min, I;
   min = ar[0];
   for(I = 1 ; I < 10 ; I++)
   {
      if(ar[I] < min)
      {
         min = ar[I];
      }
   }
   return &min;
}
```

(2 Points)

- a) -20
- b) Function can't return address
- c) No value will be printed
- d) There is a value that we don't know appear on the screen

23.What will be the output of the following C++ code?

```
#include <iostream.h>
```

```
void swap(int &a, int &b);
 int main() {
 int a = 5, b = 10;
 swap(a, b);
 cout << "In main " << a << b;
 return 0;
}
void swap(int &a, int &b) {
int temp;
 temp = a;
a = b;
 b = temp;
cout << "In swap " << a << b;
 }
```

(2 Points)

- a) In swap 105 In main 105
- b) In swap 105 In main 510
- c) In swap 510 In main 105
- d) In swap 510 In main 510

24.What is the difference between references and pointers?

(2 Points)

- a) References are an alias for a variable whereas pointer stores the address of a variable
- b) References and pointers are similar
- c) References stores address of variables whereas pointer points to variables
- d) Pointers are an alias for a variable whereas references stores the address of a variable

25.

```
#include
int f(int &x, int c)
{
c = c - 1;
if (c == 0)
return 1;
x = x + 1;
return f(x, c) * x;
}
int main()
{
int a = 4;
cout<<f(a,a);
```

(2 Points)

- a) 343
- b) 336
- c) 120
- d) 840

26.What will be the output of the following C++ code?

```
#include <iostream.h>
struct Time {
int hours;
int minutes;
int seconds;
};
int toSeconds(Time now);
int main() {
Time t; t.hours = 5;
t.minutes = 30;
t.seconds = 45;
cout << "Total seconds: " << toSeconds(t) << endl;
return 0;
}
int toSeconds(Time now) {
return 3600 * now.hours + 60 * now.minutes + now.seconds;
}
```

(2 Points)

- a) 19845
- b) 20000
- c) 15000
- d) 19844

27.

```
#include <iostream>
class Person
{
    protected:
        int id;
    public:
        Person(int i)
        {
            id = i;
        }
        Person(Person &p)
        {
            id = p.id;
        }
        int GetId()
        {
             return id;
        }
        void SetId(int m)
        {
              id = m;
        }
};
```

```
class Employee: public Person
{
      int salary;
   public:
      Employee(int i, int s) :Person(i)
      {
         salary = s;
      }
      Employee(Employee &e)
      {
         salary = e.salary;
      }
      int GetSalary()
      {
         return salary;
      }
      void SetSalary(int m)
      {
         salary = m;
      }
};

Employee FillEmp()
{
   Employee em(3, 4);
   return em;
}

int main()
{
   Employee emp(7, 8);
   emp = FillEmp();
   cout << emp.GetId() ;
   cout << emp.GetSalary() << endl;
   return 0;
}
```

What is the output of the previous program?
(2 Points)

- 34
- 78
- Compile error There is no default constructor
- Non of the above

28.What will be the output of the following C++ code?

```
 #include <iostream.h>
int main () {
int n; for (n = 5; n > 0; n--)
```

```
{
cout << n;
 if (n == 3) break;
 }
 return 0;
 }
```

(2 Points)

- a) 543
- b) 54
- c) 5432
- d) 53

29.What will be the output of the following C++ code?

```
#include <iostream.h>
void Sum(int a, int b, int & c)
{ a = b + c;
 b = a + c;
c = a + b; }
 int main() {
 int x = 2, y =3; Sum(x, y, y);
 cout << x << " " << y;
return 0;
}
```

(2 Points)

- a) 2 3
- b) 6 9
- c) 2 15
- d) compile time error

30.What will be the output of the following C++ code?

```
 #include <iostream.h>
int func(int m = 10, int n)
```

default parameter must be the last parameter

```
{ int c;
c = m + n;
return c;
}
 int main() {
cout << func(5, 6);
return 0;
}
```

(2 Points)

- a) 15
- b) 10
- c) compile time error
- d) 30

31.What will be the output of the following C++ code?

```
#include <iostream.h>
int func(int m , int n = 10)
{ int c;
 c = m + n;
 return c;
}
Int func(int a)
{
return (a*a);
}
int main()
{ cout << func(5);
 return 0;
}
```

(2 Points)

- a) 15
- b) 11
- c) compile time error
- d) 30

32.What will be the output of the following C++ code?

```
 #include <iostream.h>
class rect {
int x, y;
public:
void val (int, int);
int area () {
return (x * y);
}
};
void rect::val (int a, int b)
 {
 x = a; y = b;
}
 int main ()
{
rect Rect; Rect.val (3, 4);
 cout << "rect area: " << Rect.area();
return 0;
 }
```

(2 Points)

- a) rect area: 24
- b) rect area: 12
- c) compile error
- d) rect area: 56

33.What will be the output of the following C++ code?

```
#include <iostream.h>
 class CDummy
{
 public:
int isitme (CDummy& param);
};
int CDummy::isitme (CDummy& param)
{ if (&param == this)
return 1;
else
return 0;
}
int main () {
 CDummy a;
 CDummy *b = &a;
 if (b->isitme(a))
{
cout << "execute";
 } else
{
cout<<"not execute";
}
return 0;
}
```

(2 Points)

- a) execute
- b) not execute
- c) error
- d) both execute & not execute

34.Which of the following, when used as the /* body */ of method sum, will enable that method to compute 1 + 2 + ⋯+ n correctly for any n > 0?

//return 1 + 2 + ... + n

```
int sum(int n)
{
/* body */
}
```

I)
```
return n + sum(n - 1);
```

II)
```
if (n == 1)
        return 1;
    else
        return n + sum(n - 1);
```

III)
```
if (n == 1)
        return 1;
    else
         return sum(n) + sum(n - 1);
```

(2 Points)

- a) I only
- b) II only
- c) III only
- d) I and II only
- e) I, II, and III

Never give out your password.Report abuse
This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.
Powered by Microsoft Forms
|
The owner of this form has not provided a privacy statement as to how they will use your response data. Do not provide personal or sensitive information.
| Terms of use

OOP using C++ Exam Duration : 90 Min.

* Required

1. Name *

Enter your answer

2. Track *

Open Source

3. class Point

```
{
 float x, y;
Point (float a, float b) { x=a; y=b;}
Point () { x=0; y=0; }
}
```

To write copy constructor to class point, what would be its signature?
(3 Points)

- ○ Point (Point)
- ◉ Point (Point &)
- ○ Point & Point (Point &).
- ○ Point & Point (Point)

- void Point (Point)
- void Point (Point &)
- None of the above

4. class Nice

```
{   int a ;
      public:
            Nice( ) { a = 0 ; }
            Nice(Nice & myN)
                {   this -> a =  myN.a ;
                    cout<<"I am the copy constructor  " ;
                }
            void setA(int m)  { a = m; }
             int getA()   { return a ; }
};
void show(Nice &obj)
{
cout<<"I am the show function, value is: " << obj.getA() ;
}
void main()
{   Nice n1;
    n1.setA(15) ;
    show(n1) ;
}
```

(3 Points)

- (●) I am the show function, value is: 15.
- ( ) I am the show function, value is: 15. I am the copy constructor.
- ( ) I am the copy constructor. I am the show function, value is: 15.
- ( ) I am the copy constructor.

5. The Destructor of a class is responsible for removing the object from the memory .

(1 Point)

- (●) True

- False

6. When overloading a certain function, it is obligatory that we specify a different number of parameters for the new function.
(1 Point)

- True
- False

7. What will be the output when you compile and run the following piece of code?

```
class Test
{
        int x;
         public:
            Test( ) {  x = 0;   }
            Test(int y) {  x = y++; }
            Test(Test &r)   {  x = ++r.x;  }
             void print( )      {cout<<x ;  }
  };
  void main()
   {
                Test t(1) ;
                 t.print() ;
                 Test x(t) ;
                 x.print() ;
                 t.print() ;
    }
```

(3 Points)

- 121
- 122
- 222
- 233

8. If we did not specify a constructor to the class, then :
(3 Points)

- ○ we won't be able to create object of class
- ○ we won't be able to create object of class, and compiler will give compilation error
- ○ we won't be able to create object of class, and compiler will give warning
- ○ it will generate run-time error
- ◉ None of the above

9. We can overload Destructor in the class
(1 Point)

- ○ True
- ◉ False

10. Which of the following statements are true about constructor?
(3 Points)

- [x] A constructor can be overloaded.
- [x] A constructor is a special member function with the same name of the class.
- [ ] A constructor can return a primitive or an object reference.
- [ ] All the above

11. What will be the output when you compile and run the following piece of code?

```
class Parent
  {
    protected:
            int x;
    public:
            Parent(int m)
                    { x = m ; }
     friend void display( );
   };
class Child : public Parent
  {
    private:
            int y;
     public:
          Child(int m, int n) : Parent(m)
           {  y = n ;  }
   };
  void display ()
          {
            Child c(3,4);
            cout <<"x="<<c.x<<"y="<<c.y; // Line 1
           }

   void main ()
          {
            display();
           }
```

(3 Points)

- ○ Compilation Error at Line 1, Child::x is inaccessible
- ○ Compilation Error at Line 1, Child::y is inaccessible
- ◉ A and B
- ○ The code compiles successfully.

12. Multilevel Inheritance is allowed as an object Oriented concept, but it is not allowed
    in C++;

(1 Point)

- ○ True
- ◉ False

13.) In order for the following piece of code to compile successfully, what are the constructors that are expected to exist in the Base class?

```
class Child : public Base
        {
                public:
                    Child(int x)   { }
                       Child(int x, int y) : Base(x,y) { }

        };
```

(3 Points)

- ◉ Base( ) and Base(int , int).
- ○ Base( ) and Base(int).
- ○ Base(int) and Base(int , int).
- ○ Base(int , int).

14.

```
class Super
{
    protected:
                Super(int a)
                 {
                      this.a = a;     // Line 1
                 }
   private:                          // Line 2
              int a;
};
class Sub: public  Super
{
      public:
                 Sub(int a):Super(a)  { }
      public:
              Sub()                 //Line 3
                {
                this.a= 5;          //Line 4
              }
};
```

Which steps will allow Sub to compile?
(3 Points)

- [ ] Class Sub compile successfully.
- [ ] Comment Line 2
- [ ] In Line 1 and 4 to, use (*this).a instead of this.a;
- [ ] Change Line 1 and 4 to, this(a);
- [ ] Change Line 3 to, Sub(): Super(5)
- [ ] Change Line 3 to, Sub(): this(5)
- [ ] All the above.

15. What will be the output when you compile and run the following piece of code?

```
class Base
        {       public:
                    Base() { cout<<"Welcome     " ; }
        };
class Derived : public Base
        { public:
                Derived() { cout<<"Hello     " ; }
        };
void main()
      {  Base myBase ;
         Derived myDerived ;
      }
```

(3 Points)

- ○ Welcome Hello
- ○ Hello Welcome
- ○ Welcome Hello Welcome
- ◉ Welcome Welcome Hello

16. In order to turn a class into an abstract class, which of the following do we need to do?

(3 Points)

- ○ Write the abstract keyword before the name of the class.
- ○ Make the class a pure virtual class.
- ◉ Write one or more pure virtual functions inside the class.
- ○ A and C.
- ○ None of the above

17. If a certain function is made friend for class A, then that function can access only the private members of class A.
(1 Point)

- ○ True
- ◉ False

18. A protected member of a class can be directly accessed by its name inside another class if and only if that other class is a child of that class.
(1 Point)

- ◉ True
- ○ False

19. Which of the following most closely describes the process of overriding?
(3 Points)

- ○ A class with the same name replaces the functionality of a class defined earlier in the hierarchy.
- ◉ A function with the same name replaces the functionality of a function defined earlier in the inheritance hierarchy.
- ○ A function with the same name but different parameters gives multiple uses for the same function name.
- ○ Making a class abstract so that no objects can be declared from it.

20. Assume you have a class M that contains a pointer to an object of class N. Assume that we declare an object of M in the main( ) function. When will the body of the constructor of class N be executed?
(3 Points)

- ◉ When any member function of the class M is called.

- ◯ After the body of the constructor of class M is executed.
- ◯ Before the body of the constructor of class M is executed.
- ◯ None of the above.

21.
```
class Parent
    { public:
            int x;
            Parent(int m)   { x = m ; }
    };
class Child : protected Parent
    {
        public:
            int y;
            Child(int m, int n) : Parent(m) {   y = n ;  }
    };
class GrandChild : public Child
    {    int z ;
        public:
            GrandChild(int a, int b, int c) : Child(a,b) {  z = c ; }
    };
void main( )
    {  GrandChild obj(3,5,7);
       cout<<"Value of x is: "<<obj.x <<endl ;          //Line 1
       cout<<"Value of y is: "<<obj.y <<endl ;          //Line 2
       cout<<"Value of z is: "<<obj.z <<endl ;          //Line 3
    }
```

(3 Points)

- [x] Compiler Error at Line 1
- [ ] Compiler Error at Line 2
- [x] Compiler Error at Line 3
- [ ] The code compiles successfully.

22. What will be the output when you compile and run the following piece of code?

```
class Tester
 {   public:
                int x ;
               static int var ;
                Tester(int a) {   x = a ; }
               static void myFunction(int a)
                   {
                   Tester obj(9) ;      // Line 1
                   obj.x=a ;            // Line 2
                      cout<<obj.x;
                   }
  };
  int Tester::var=0;
 void main()
 {
   Tester myT(7) ;
   Tester::myFunction(15);
   cout<<myT.x ;
   cout<<Tester::var<<endl ;       // Line 3
  }
```

(3 Points)

- (●) 15 7 0
- ( ) 7 15 0
- ( ) 15 9 0
- ( ) 9 15 0
- ( ) Compilation error at line 1
- ( ) Compilation error at line 2
- ( ) Compilation error at line 3
- ( ) None of the above.

23. The term " Composition " refers to an object of a class that contains another object
of another class inside it.
(1 Point)

- (●) True
- ( ) False

24. Which of the following is true about an object member function?
(4 Points)

- [ ] It can be called using the name of the class.
- [x] It can access static variables of the class.
- [x] It has a "this" pointer as an implicit parameter passed to it.
- [ ] It can access the instance variables.
- [ ] It cannot be overloaded.
- [ ] It can call other member functions from inside it.

25. Assume you have a member function with the following prototype?
void myFunc(int x) ;
Which of the following are valid ways to overload it?
(3 Points)

- [x] void myFunc(char ch) ;
- [ ] int myFunc(int x) ;
- [x] void myFunc(char c1, char c2) ;
- [ ] float myFunc(int x, int y) ;

26. Which of the following is true about the function prototype below?
    void myFunc (int myDef=17, int myVar , int myNormalVar=5) ;
    (3 Points)

- (●) We should also give a default value to myVar.
- ( ) We must only give a default parameter for myNormalVar and not the others.
- ( ) The function is correct in that way.

27. What does the following piece of code do?

```
void main( )
        {
         float *ptr ;
          ptr = new float[15];
        }
```

(3 Points)

- ( ) Allocate space for a float variable that is not initialized
- ( ) Allocate space for an array of 15 float elements that are not initialized
- ( ) Allocate space for an array of 15 float elements that is initialized by the value 0
- ( ) Allocate space for an array of 15 float elements where all the elements are initialized by the value 15
- (●) Compiler Error.

28. Static member variable can only be modified through static member functions.
    (1 Point)

- ( ) True
- (●) False

29. "A plane is a machine that has a motor and has wings".
"A refrigerator is a machine that has a motor and has shelves".
Which of the following best describes the previous statements as a set of classes?
(3 Points)

- ○ 1 class: A machine class that has an attribute for the type of machine.
- ● 2 classes: A plane class that has two attributes, and a refrigerator class that also has two attributes.
- ○ 3 classes: A machine class that has one attribute: motor. A plane class that inherits from the machine class. And a refrigerator class that inherits from the plane class.
- ○ 3 classes: A machine class that has one attribute: motor. A plane class that inherits from the machine class. And a refrigerator class that also inherits from the machine class.

30.
```
class Stack
    {
        int tos,size;
        int * st;
        Stack( int s=5) { tos=0;size=s; st=new int[size];}
        ~ Stack( ){ delete []st;}
    };
void main()
{
  Stack s1(20);
}
```

what shall we add to the above class Stack to declare another object s2 from class Stack where s2 is declared in terms of s1; Stack s2(s1)?
(3 Points)

- ○ We must specify overload of assignment operator for class Stack.
- ○ We must define a copy constructor to class Stack
- ● A and B
- ○ This situation cannot be achieved in C++, however, it has been solved in other programming languages.

31. What will be the output when you compile and run the following piece of code?

```
class Parent
 {
     int x;
     Parent(int m){ x = m ; }
 };
class Child : public Parent
 {  public:
            int y;
            Child(int m, int n) : Parent(m) //Line 1
           {   y = n ;  }
 };

void main( )
{
    Child obj(3,5,7);          //Line 2
   cout<<"Value of x is: "<<obj.x <<endl ;  //Line 3
   cout<<"Value of y is: "<<obj.y <<endl ;  //Line 4
}
```

(3 Points)

- [x] Compiler Error at Line 1
- [x] Compiler Error at Line 2
- [x] Compiler Error at Line 3
- [ ] The code compiles successfully

32. When inheriting from a Base class, the Derived class will inherit only the protected, and public members of the Base class.

(1 Point)

- (●) True
- ( ) False

33. The "this" pointer is an array of pointers to all the objects created from a class

(1 Point)

- (●) True
- ( ) False

34. What will be the output when you compile and run the following piece of code?

```
class Parent
        {
         protected:
            int myVar;
         public:
            Parent(int x) { myVar=x; }
            void powerTwo(){ cout<<myVar*myVar; }
            virtual void powerThree() { cout <<myVar*myVar*myVar; }
        };
    class Child:public Parent
        {
         protected:
             int myData;
        public:
             Child(int a, int b) : Parent(a) {myData= b;}
            void powerTwo() { cout <<myData*myData; }
            void powerThree() { cout<<myData*myData*myData; }
        };
    void main()
    {
       Child myCh(2,3);
       Parent *myPtr;
      myPtr = &myCh;
      myPtr->powerTwo(); //Line1
      myPtr->powerThree(); //Line 2
}
```

(3 Points)

- ( ) 4 8
- (●) 4 27
- ( ) 9 27
- ( ) 9 8
- ( ) Compilation Error at Line1

- ◯ Compilation Error at Line2

35. What will be the output when you compile and run the following piece of code?

```
class Parent
        {  int y;
           static int z;
           public:
                    Parent( )
                        {
                           z=0;   // Line1
                        }
                    Parent (int a=5)   //Line 2
                        {
                            y=a;
                        }
        };
  void main( )
        {
            Parent d(4);     //Line 3
            Parent m;        //Line 4
        }
```

(3 Points)

- ◯ Compilation Error at Line 1, an object member function cannot access a static member
- ◯ Compilation Error at Line 2, constructor should initialize static member (z=0;)
- ◯ Compilation Error at Line 3
- ◉ Compilation Error at Line 4.
- ◯ The code compiles successfully.

Submit

Never give out your password. Report abuse

This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.

Powered by Microsoft Forms | Privacy and cookies | Terms of use

Ministry of Communication and Information Technology Information Technology Institute	December, 2007 1:15 Hours Open Book Exam

Q. 1 True or False

1. It is not recommended to make the function inline when the function is return value . (F)
2. The child must inherit all parent member. (F) X
3. The class contains pure virtual function is called abstract class (T)
4. The virtual function must be overridden in any Derived class. (F)
5. A pointer to a base class can point to objects of a derived class. (T)

Q. 2 Select the correct answer(s)

1. A constructor is:

a) A member function initialize object in memory.
b) A member function called automatically when an object is being created in the memory.
c) A private member function can never be called.
d) None of the above.

2. Consider the following code:

```
void main()
{
   int *p;
   p=new int[5];
}
```

The above code:

a) Allocate for an integer that is not initialized.
b) Allocate for an array of integers of 5 elements that is not initialized.
c) Allocate for an integer that is initialized by 5.
d) Allocate for an array of integers where all its elements is initialized by 5.

Base

3. The constructor of derived class called automatically:

a) When any member function of the derived class called.
b) After the constructor of the derived class started.
c) Before the constructor of the derived class started.
d) We do not know when exactly will be started.

5. Virtual functions allow you to

a. create an array of type pointer-to-base class that can hold pointers to derived classes.
b. create functions that can never be accessed.
c. group objects of different classes so they can all be accessed by the same function code.
d. use the same function call to execute member functions of objects from different classes.

6. A static function

e. should be called when an object is destroyed.
f. is closely connected to an individual object of a class.
g. can be called using the class name and function name.
h. is used when a dummy object must be created.

7. **The protected members access specifier for the class user is equivalent to:**
   a) public member
   b) private member
   c) global variable
   d) local variable

Q. 3 Design and implement the following functions using C Programming Language

In the Geometry Science, the Circle, Triangle, Rectangle are Shapes. Each Shape consists of 2 dimensions, and the area of each shape is defined by the following formula : "Area = dimension1 * dimension2 * const".
The const depends on the Shape itself

1. Extract the classes from the last sentences
2. state their relation between them
3. Design and Implement a new class named PICTURE that consists of 5 shapes (Circle and/or Triangle and/or Rectangle)

****

Best Wishes

1- Extract the classes:
Circle
Triangle
Rectangle
Shape (abstract base class)

2-State their relation between them:
Circle, Triangle, and Rectangle are shapes (inheritance).
Picture has an association with instances of Circle, Triangle, and/or Rectangle (association).

```
#include <iostream>

class Shape
{
  protected:
    double dim1,dim2;

  public:
      Shape()
      {
        dim1=dim2=0;
      }
      Shape(double x)
      {
        dim1=dim2=x;
      }
      Shape(double x, double y)
      {
        dim1=x;
        dim2=y;
      }
      void SetDim1(double x)
      {
        dim1=x;
      }
      void SetDim2(double y)
      {
        dim1=y;
      }
      double GetDim1()
      {
        return dim1;
      }
      double GetDim2()
      {
        return dim2;
      }
      virtual double GetArea()=0;
};

class Circle: public Shape
{
  public:
    Circle(){}
    Circle(double radius):Shape(radius){}
    double GetArea()
    {
      return (3.14*dim1*dim2);
    }
};

class Triangle: public Shape
{
  public:
    Triangle(){}
    Triangle(double x):Shape(x){}
    Triangle(double base, double height):Shape(base,height){}
    double GetArea()
    {
      return (0.5*dim1*dim2);
    }
};

class Rectangle: public Shape
{
  public:
    Rectangle(){}
    Rectangle(double x):Shape(x){}
    Rectangle(double x, double y):Shape(x,y){}
    double GetArea()
    {
      return (2*(dim1+dim2));
    }
};

class Picture
{
  Shape* shapes[5];
  public:
    Picture(Shape* shs[5])
    {
      for(int i=0;i<5;i++)
      {
        shapes[i]=shs[i];
      }
    }
    //extra
    double GetTotalArea()
    {
      double sum=0;
      for(int i=0;i<5;i++)
      {
        sum+=shapes[i]->GetArea();
      }
      return sum;
    }
};

int main()
{
  Circle c(2);
  Triangle t(2,4);
  Rectangle r(6,8);
  Shape* shapes[5]={&c,&c,&t,&r,&t};
  Picture p(shapes);
  cout << p.GetTotalArea()  << endl;
  return 0;
}
```

Object –Oriented Programming Final Exam.

Q1: True or False.

1. Any static member can access all static and instance members. (F)
2. You can have more than a constructor in the same class. (T)
3. A destructor is a member function that removes object from memory. (F)
4. An abstract class is a class that contains any virtual functions. (F)
5. You can use more than a template for one class. (T)
6. A constructor is always declared void because it doesn't return a value. (F)
7. Aggregation and association are types of relationships between a base class and derived classes. (F)
8. The embedded object is an object declared with default constructor. (F)
9. You cannot overload the scope operator (::). (T)
10. You can define a copy constructor in class. (T)

Q2: Select the correct answer(s).

1. Object-Oriented is used because: (2 answers)
   - The code is smaller.
   - The program is faster.
   - The code is reusable.
   - The code is easy to maintain.

2. Which of the following refers to "Hiding data inside class methods" ?
   - Abstraction
   - Encapsulation
   - Polymorphism
   - Inheritance

3. We use the inheritance for:
   - Increasing more member functions only to the base class.
   - Extending base class functionality.
   - Including the base class.

4. The following are examples of polymorphism: (2 answers)
   - Operator overloading
   - Function overriding
   - Friend functions
   - Constructor overloading.

5.
```
class XX
{
        int i;
        char c;
        void seti (int r) { }
        char getc ( ) { return c; }
};
void main( )
{
        XX x;
        x.seti(8);
}
```

On running, the previous code:

- compiles with no errors
- the compiler generates an error because the class XX cannot be instantiated because it has no public constructor.
- The linker generates an error because the class XX cannot be instantiated because it has no public constructor.
- The compiler generates an error because the method "seti" is non accessible through object x.

6.
```
void printNumber( )
{
        Int num=5;
        for (i=0 ; i<10 ; i++)
        { cout<<num; }
        int i;
        char c;
}
```

In C++, on calling the previous function:

- An error occurs because i & c cannot be declared after the loop (invalid declaration).
- An error occurs because the var i is not visible in the for loop (undefined symbol i)
- No error occurs.

7. The constructor of the base class called automatically:
   - When any member function of the derived class called.
   - After the constructor of the derived class started.
   - Before the constructor of the derived class started.
   - We do not know when exactly it will be started.

8. class X
```
{
        int k;
        public:
                void setK(int k) {……..}
};
```
Which can best be written in the space?
- this->k=k;
- k=k;
- this.k=k;

9. class Person
```
{
        public:
                virtual void talk( )==0;
};
class Adult : public Person
{
        public:
                void talk() { cout<<"Hello"; }
};
class Female : public Person
{
        public:
                void talk( ) { cout<<"Hi"; }
};
void main( )
{
        Person p;
        Adult a;
        Female f;
}
```
The previous code:
- Compiles successfully.
- Produces an error.

10. class Child : public Base
```
{
        public:
                Child(int x) { }
                Child(int x,int y) : Base(x,y)
                { }
};
```

What are the expected constructors that must exist in the base class?

- Base( ), Base(int,int)
- Base( ), Base(int)
- Base(int), Base(int,int)
- Base(int,int)

11. The class that contains a pure virtual function is called:
    - Base class.
    - Derived class.
    - Abstract class.
    - Embedded class.

12.
```
#include <iostream.h>

class XX
{
        int i;
        char c;

        public:
                void seti(int r) { }
                void getc( ) { return c; }
                void printHi( ) { cout<<"Hi"; }
};
void printHi( ) { cout<<"Hello"; }
void main( )
{
        XX x;
        x.printHi( );
}
```
The previous code prints:
    - Hello.
    - Hi.
    - Hi then Hello
    - None of the above, it generates an error.

13.
```
class Student
{
        char name[20];
        public:
                Student( ) { strcpy(name, "Anonymous"); cout<<name; }
                Student(char* nm) { strcpy(name,nm); cout<<name; }
                ~Student( ) { }
```

```
};
void main( )
{
        Student s1(Mona);           ……1
        Student *s2;                ……2
        S2 = new Student("Ali");    ……3
        delete s1;                  ……4
        delete s2;                  ……5
}
```

In the previous code:

- It compiles successfully.
- Line 4 generates an error.
- Line 5 generates an error.
- Both lines 4,5 generate errors.

14.
```
class Test
{
        public:
                void print(int i)
                { cout<<"int version"; }
                void print(char *c)
                { cout<<"char version"; }
};

void main( )
{
        Test t;
        char ch='p';
        t.print(ch);
}
```

Which of the statements below is true?

- Line 4 will not compile, because void methods cannot be overloaded.
- Line 7 will not compile, because there is no version of print that takes a char.
- The code will compile and produce the following output: int version.
- The code will compile and produce the following output: char version.

15.
```
class Base
{
        public:
                int x;
};

class Child : private Base
{
        public:
                int y;
};

void main( )
{
        Child c;
        c.x=10;
        cout<<"x="<<x;
}
```

What happens in program?

- Compiler error because x is not declared in Child class.
- Compiler error because of the Child inheritance from private class.
- x=10
- Runtime error.

Q3: Design Question.

Design a set of classes to calculate the volume of the following shapes:

Cylinder	($\pi r^2 l$), where r: radius, l: length.
Cube	(l^3), where l: length.
Box	(w*h*l), where w: width, h: height, l: length.

Answers:

Q1:
1. F
2. T
3. F
4. F
5. T
6. F
7. F

8. F
9. T
10. T

Q2:
1. c, d
2. b
3. b
4. a
5. d
6. b
7. c
8. a
9. b
10. a
11. c
12. d
13. b
14. c
15. b

Q3:

```
class Shape
{
        protected:
                int length;
        public:
                Shape( ) { length=0; }
                Shape( int l ) { length=l; }
                virtual void calc_volume( )=0;
};
class Cylinder : private Shape
{
        private:
                int radius;
        public:
                Cylinder( ) : Shape( )
                { radius = 0; }
                Cylinder (int l,int r) : Shape(l)
                { radius = r; }
                void calc_volume( )
                {
                        float result;
                        result = (22/7)*radius*radius*length;
                        cout<<"Cylinder volume= "<<result<<endl;
```

```
                }
};
class Box : public Shape
{
        protected:
                int width,height;
        public:
                Box( ) : Shape( )
                { width = height = 0; }
                Box(int l, int w, int h) : Shape(l)
                {
                        width = w;
                        height = h;
                }
                void calc_volume( )
                {
                        int result;
                        result = width * height * length;
                        cout<<"Box volume= "<<result<<endl;
                }
};
class Cube : private Box
{
        public
                Cube( ) : Box( ) { }
                Cube(int w) : Box(w,w,w)
                { }
};
```

OOP exam - Assiut PWD ...

True or False (10 marks)

2

The destructor of a class is responsible for removing the object from the memory
(1 Point)

- True
- False

3

Multilevel Inheritance is allowed as an object Oriented concept, but it is not allowed in C++
(1 Point)

- True
- False

4

Association is a special type of Composition
(1 Point)

- True
- False

5

The term "Encapsulation" refers to an object of a class that contains another object of another class inside it
(1 Point)

- True
- False

6

When inheriting from a Base class, the Derived class will inherit only the protected, and public members of the Base class
(1 Point)

- True
- False

7

Static member variable can only be modified through static member functions
(1 Point)

- True
- False

8

If a certain function is made friend for class A, then that function can access only the private members of class A
(1 Point)

- True
- False

9

Composition is special type of Inheritance; composition is stronger relation between 2 objects than inheritance
(1 Point)

- True
- False

Inheritance establishes an "is-a" relationship, while composition establishes "Has-a" relationship

10

The "this" pointer is an array of pointers to all the objects created from a class
(1 Point)

- True
- False

11

When overloading a certain function, it is obligatory that we specify a different number of parameters for the new function
(1 Point)

- True
- False

int myfunction(int a , char c)
int myfunction(char a , int c)
changing their data type or order also overloads them

Back | Next | Page 2 of 3

Never give out your password. Report abuse

This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.

Powered by Microsoft Forms | Privacy and cookies | Terms of use

Intake 42 OOP Exam

Duration : 90 Min. (35 Questions)

Regards

1 Full Marks in English !

Take new contest

---

1 / 35

Code :

```
class A
{
    static void
    {
        System.out.
    }
}
```

1. What does the following piece of code do?

- Compiler error
- Run fail
- Run pass

2. Which of the following is true?

- Static keyword can be used with class in C, C++, C#, Java
- Static keyword can be used only in C# and Java because it is not supported by the other two
- Static keyword can be used only in C# because other two are not supported by the other two
- None of the above

3. Which of the following statements are true?

- Static keyword can be used with class, variable, method and block
- Static keyword can be used with class, variable, method and block in C# and Java
- Static keyword can be used with class, variable, method and block in C# and C++
- None of the above

4. Which of the following statements are true about constructors?

- Constructor can be used to initialize objects
- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- None of the above

5. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

6. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

7. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

8. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

9. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

10. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

11. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

12. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

13. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

14. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

15. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

16. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

17. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

18. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

19. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

20. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

21. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

22. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

23. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

24. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

25. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

26. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

27. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

28. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

29. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

30. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

31. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

32. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

33. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

34. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

35. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

---

1 / 35

Code :

```
class A
{
    protected
    {
        protected
        {
            Console.Write("");
        }
    }
    public
    {
        Console.Write("");
    }
}
class Sub : public Super
{
    public
    {
        Console.Write("");
        base.Sub();
        base.Sub(int, int);
        base.Sub(int, int, int);
        base.Sub(int, int, int, int);
        base.Sub(int, int, int, int, int);
        base.Sub(int, int, int, int, int, int);
        base.Sub(int, int, int, int, int, int, int);
        base.Sub(int, int, int, int, int, int, int, int);
    }
}
static void
{
    Console.Write("");
    Console.Write("");
    Console.Write("");
    Console.Write("");
    Console.Write("");
    Console.Write("");
    Console.Write("");
    Console.Write("");
}
}
```

1. What will be the output?

- Compiler error
- Run fail
- Run pass

2. Which of the following is true?

- Static keyword can be used with class in C# and Java
- Static keyword can be used with class in C# and C++
- Static keyword can be used with class in C# and C++
- None of the above

3. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

4. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

5. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

6. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

7. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

8. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

9. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

10. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

11. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

12. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

13. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

14. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

15. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

16. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

17. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

18. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

19. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

20. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

21. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

22. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

23. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

24. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

25. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

26. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

27. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

28. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

29. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

30. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

31. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

32. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

33. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

34. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

35. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

---

1 / 35

1. In the Syntax "Computer" refers to an object of a class that contains a pointer to another object.

- True
- False

2. Which of the following is true?

- Static keyword can be used with class in C# and Java
- Static keyword can be used with class in C# and C++
- Static keyword can be used with class in C# and C++
- None of the above

3. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

4. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

5. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

6. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

7. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

8. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

9. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

10. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

11. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

12. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

13. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

14. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

15. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

16. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

17. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

18. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

19. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

20. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

21. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

22. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

23. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

24. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

25. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

26. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

27. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

28. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

29. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

30. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

31. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

32. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

33. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

34. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

35. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

---

1 / 35

1. Which of the following is true about the function prototype Sales?

- Function prototype Sales is a function with no parameters
- Function prototype Sales is a function with two parameters
- Function prototype Sales is a function with three parameters
- None of the above

2. Which of the following is true about the function prototype Sales?

- Function prototype Sales is a function with no parameters
- Function prototype Sales is a function with two parameters
- Function prototype Sales is a function with three parameters
- None of the above

3. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

4. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

5. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

6. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

7. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

8. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

9. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

10. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

11. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

12. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

13. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

14. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

15. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

16. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

17. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

18. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

19. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

20. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

21. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

22. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

23. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

24. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

25. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

26. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

27. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

28. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

29. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

30. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

31. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

32. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

33. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

34. Which of the following statements are true about destructors?

- Destructor can be used to free memory
- Destructor can be used to free memory in C# and Java
- Destructor can be used to free memory in C# and C++
- None of the above

35. Which of the following statements are true about constructors?

- Constructor can be used to initialize static variables
- Constructor can be used to initialize static variables in C# and Java
- Constructor can be used to initialize static variables in C# and C++
- None of the above

---

1 / 35

1. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

2. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

3. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

4. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

5. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

6. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

7. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

8. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

9. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

10. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

11. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

12. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

13. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

14. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

15. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

16. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

17. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

18. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

19. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

20. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

21. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

22. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

23. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

24. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

25. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

26. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

27. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

28. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

29. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

30. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

31. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

32. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- None of the above

33. Which of the following is true about a class?

- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it is a base class) and contains a block of data.
- Class is a data type that can be directly accessed by its name (name another class if and only if it

32:45

Object-Oriented Programming using C++ Final Exam - Intake 43 (60 Minutes)

Q1: True or False

3. The instance members can deal with static members in any class.(1 Point)

- ○ True
- ◉ False

4. Function Overloading is considering as one of the ways of Polymorphism.(1 Point)

- ◉ True
- ○ False

5. The destructor is a function which removes the object from memory.(1 Point)

- ○ True
- ◉ False

6. The copy constructor is called when create an object as a complete copy from another object.(1 Point)

- ◉ True
- ○ False

32:45

7. Children classes can override the static function inherited from the parent class. (1 Point)

- ○ True
- ◉ False

8. In Object-Oriented programming, any objects from any type can receive the same messages.(1 Point)

- ○ True
- ◉ False

i don't know

9. If we do not define any constructor for a class, then the compiler creates a constructor by default, which has no input parameters.(1 Point)

- ◉ True
- ○ False

10. The operation of adding on the functionalities of an operator is called Operator overriding.(1 Point)

- ◉ True
- ○ False

11. In Object-Oriented concept, we may make multiple Inheritance.(1 Point)

- ○ True
- ◉ False

32:45

12. If the mode of inheritance is private inheritance, the inherited protected member from a parent class can be accessed directly inside a child class.(1 Point)

- (●) True
- ( ) False

13. The term "**Association**" refers to a class has an attribute from primitive data type. (1 Point)

- (●) True
- ( ) False

i don't know

14. In the same class, any instance method can deal with any instance variables declared inside the class.(1 Point)

- (●) True
- ( ) False

15. A protected member of a class can be directly accessed by another class if and only if that other class in the same file of the first class.(1 Point)

- ( ) True
- (●) False

16. The goal of making setter and getter of an attribute to simplify the accessibility of it.(1 Point)

- (●) True
- ( ) False

not sure

32:45

17. All objects that are instance of the same class can perform the same actions. (1 Point)

- ○ True
- ◉ False

18. Association is an "is-a" relationship between a class and its parent class.(1 Point)

- ○ True
- ◉ False

19. Overriding means that re-implementing of a function inherited from the base class by a function in the derived class with the same name and the same input parameters.(1 Point)

- ○ True
- ◉ False

20. C++ is a structured programming language supports Object-Oriented paradigm. (1 Point)

- ◉ True
- ○ False

21. The relation between the class ***Monkey*** and the class ***Animal*** is association relation.(1 Point)

- ◉ True
- ○ False

32:45

22. We cannot create an object from a virtual base class.(1 Point)

- (●) True
- ( ) False

Back

Never give out your password. Report abuse

This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.

Powered by Microsoft Forms |
The owner of this form has not provided a privacy statement as to how they will use your response data. Do not provide personal or sensitive information.
| Terms of use

34:35

Object-Oriented Programming using C++ Final Exam - Intake 43 (60 Minutes)

Q2. Select the most appropriate answer(s):

23. When the compiler cannot differentiate between two overloaded functions, it is called:(2 Points)

- ○ a. Overloaded.
- ○ b. Destructed.
- ○ c. Overridden.
- ◉ d. Ambiguous.

24. The Instance (non-static) variable:(2 Points)

- ○ a. It is a variable declared in the private section and can be accessed by any method in the same class
- ◉ b. It is a variable declared in any section in a class and can be accessed by any method in the same class.
- ○ c. It is a variable declared inside any function and can be accessed by any other methods in the same class
- ○ d. It is a variable declared inside any function and cannot be accessed by any other methods in the same class.

34:35

25. If you design a class that needs special initialization tasks, you will have to create a(n):(2 Points)

- (●) a. Setter method.
- ( ) b. Initializer Method.
- ( ) c. Constructor.
- ( ) d. Destructor.

26. Which of the following statements is correct related to Object-Oriented concepts? (2 Points)

- ( ) a. Stand-alone function is one of the main parts of any Object-Oriented program.
- (●) b. Every object is instantiated from a particular type.
- ( ) c. It's not necessary to make inheritance among classes.
- ( ) d. All objects from the same concrete class may perform different actions.

27. The best description for the association relation among the classes:(2 Points)

- ( ) a. When a class using object(s) from other class(es).
- ( ) b. Objects that are declared using the default constructor.
- ( ) c. An object from a class that is inherited from another class
- (●) d. b and c.

34:35

28. For a derived class from a base class, it can directly access: (2 Points)

- a. The private and protected members of the base class.
- b. Only the private members of the base class.
- c. Only the protected members of the base class.
- d. The protected and public members of the base class.

34:35

29. **consider the following code:**

```
class Base
{
      public:
           int x;
           Base(int y)
          {
               x=y;
               ++x;
           }
};
class child : public Base
{
       public:
            child(int c)
           {
                 x+=1;
           }
          void print()
          {
                cout<<"x="<<x;
          }
};

void main()
{
         child c(1);
         c.print();
  }
```

(2 Points)

- a. x=1
- b. x=2
- c. x=3
- d. x=4
- e. Compilation Error.

34:35

30. **What will be the output when you compile and run the following piece of code?**

```
int sum(int a , int b) { return a+b;}
float sum(float a , float b) { return a+b;}
void main(void)
{
        clrscr();
        cout<<"sum ="<<sum(5,9)<<"\t" ;
        cout<<"sum ="<<sum(4.3,2.7) ;
 }
```

(2 Points)

- (●) a. sum = 14 sum = 7.0
- ( ) b. Compiler Error: add( ) function should be overloaded
- ( ) c. Compiler Error: Ambiguity between 'sum (int , int)' and 'sum (float, float)'
- ( ) d. sum = 14 sum = 6

34:35

31. What will be the output when you compile the following piece of code?

```
class Test
{
        int x ;
   public:
        float z ;
        int y ;
};
void main()
{
        Test t1 ;
        t1.x = 100 ;
        t1.y = 500 ;
        t1.z = 250.43 ;
 }
```

(2 Points)

- (●) a. Compiler Error regarding the variable x.
- ( ) b. Compiler Error regarding the variable y.
- ( ) c. Compiler Error regarding the variable z.
- ( ) d. The code will compile successfully.
- ( ) e. a and b.
- ( ) f. a, b, and c

34:35

32. What will be the output when you compile and run the following piece of code?

```
class Test
{
        public:
            virtual  void say() { cout<<"Test\t"; }
};
class NewTest : public Test
{
        public:
            void say() { cout<<"NewTest\t"; }
};
void main()
  {
        Test *pb ;
        Test myParent ;
        NewTest myChild ;

        pb = & myParent ;
        pb->say() ;
        pb = & myChild ;
        pb->say() ;
    }
```

(2 Points)

- (●) a. Test NewTest
- ( ) b. Test Test
- ( ) c. NewTest NewTest
- ( ) d. NewTest Test

34:35

33. **What will be the output when you compile and run the following piece of code?**

```
class Test
{
        int x;
    public:
        Test()
        {
            x = 0;
        }
        Test(int y)
        {
            x = y++;
        }
        Test(Test &r)
        {
            x = r.x++;
        }
        void print()
        {
            cout<<x ;
        }
};
void main()
{
    Test t(1) ;
    t.print() ;
    Test x(t) ;
    x.print() ;
    t.print() ;
}
```

(2 Points)

- ○ 121
- ● 112
- ○ 222
- ○ 122

34:35

34. **What will be the output when you compile and run the following piece of code?**

```
class Base
{
        public:
            Base() { cout<<"Welcome"<<"\t" ; }
 };
class Derived : public Base
{
        public:
            Derived() { cout<<"Hello"<<"\t" ; }
};
  void main()
 {
     Derived myD ;
 }
```

(2 Points)

- a. Hello
- b. Hello Welcome
- c. Welcome Hello
- d. Welcome

34:35

35. **What does the following piece of code do?**

```
void main()
{
    float *ptr ;
    ptr = new float(15);
}
```

(2 Points)

- a. Allocate space for 15 float contiguous variables that are initialized by 0.
- b. Allocate space for 15 float contiguous variables that are not initialized
- c. Allocate space for a float variable that is initialized by the value 15.
- d. Allocate space for a float variable that is not initialized.

36. In any abstract class, it has to include:(2 Points)

- a. At least a member function.
- b. At least a virtual function
- c. At least a static function.
- d. At least a pure virtual function
- e. All of the above

34:35

37. Which of the following refers to "Hiding data behind class methods"?(2 Points)

- (●) a. Abstraction.
- ( ) b. Encapsulation
- ( ) c. Polymorphism
- ( ) d. Inheritance.

38. The body of the destructor of the base class is executed automatically:(2 Points)

- (●) a. When any member function of the derived class is called.
- ( ) b. After the body of the destructor of the derived class is executed.
- ( ) c. Before the body of the destructor of the derived class is executed.
- ( ) d. We cannot determine exactly when it will be executed.

39. ***'A car is a transportation mean that has a Motor, color and 4 wheels'.***
'A bicycle is a transportation mean that has a handlebar, color and 2 wheels.

Which of the following best describes the previous statements as a set of classes? (2 Points)

- (●) a. 2 classes: A car class that has two attributes: color, motor and numberOfWheels, and a bicycle class that also has two attributes: color, handlebar and numberOfWheels.
- ( ) b. 3 classes: A Transportation_Mean class that has one attribute: color. A car class that inherits from the Transportation_Mean class and has two attributes: motor and numberOfWheels. And a bicycle class that also inherits from the Transportation_Mean class and has two attributes: handlebar and numberOfWheels.
- ( ) c. 1 class: A Transportation_Mean class that has an attribute for the: typeOfTransport.
- ( ) d. 3 classes: A Transportation_Mean class that has one attribute: numberOfWheels. A car class that inherits from the Transportation_Mean class and has two attributes: motor and color. And a bicycle class that inherits from the car class and has one attribute: handlebar.

34:35

40. The following are examples of Object-Oriented advantages: (choose all that applies)(2 Points)

- [ ] a. Simplicity
- [x] b. Reusability
- [x] c. Encapsulation
- [x] d. Modularity
- [x] e. Abstraction

41. The Embedded object is:(2 Points)

- ( ) a. An object declared with default constructor.
- ( ) b. An object has function overloading.
- (•) c. An object from a class inherited from another class.
- ( ) d. An object included by another object.

34:35

42. Given the class below, what changes do we need to make to the class in order to have a facility to add the values of an object's data to another?

```
class Test
{
        int x ;
        int y ;
    public:
        Test( )
        {
            x = 0 ;
            y = 0 ;
        }
        void setXY(int a , int b)
        {
            x = a ;
            y = b ;
        }
 };
```

(2 Points)

- a. Do operator + overloading.
- b. Declare a third temporary member integer.
- c. Create a new member function: void Add( ), that takes no parameters.
- d. All of the above.

Back

Never give out your password. Report abuse

This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.

34:35

| Terms of use

True Or False

1. Any static member can access all static and instance members. (F)
2. Function Overloading is considering as one of the ways of Polymorphism.(T)
3. The destructor is a function which removes the object from memory (F)
4. The copy constructor is called when create an object as a complete copy from another object (T)
5. Children classes can override the static function inherited from the parent class. (F)
6. In Object-Oriented programming, any objects from any type can receive the same messages (F)
7. if we do not define any constructor for a class, then the compiler creates a constructor by default, which has no input parameters (T)
8. The operation of adding on the functionalities of an operator is called Operator overriding (F)
9. In Object-Oriented concept, we may make multiple Inheritance. (T)
10. If the mode of inheritance is private inheritance, the inherited protected member from a parent class can be accessed directly inside a child class. (T)
11. The term "Association" refers to a class has an attribute from primitive data type. (F)
12. In the same class, any instance method can deal with any instance variables declared inside the class. (T)
13. A protected member of a class can be directly accessed by another class if and only if that other class in the same file of the first class (F)
14. The goal of making setter and getter of an attribute to simplify the accessibility (T)
15. All objects that are instance of the same class can perform the same actions. (T)
16. Association is an "is-a" relationship between a class and its parent class. (F)
17. Overriding means that re-implementing of a function inherited from the base class by a function in the derived class with the same name and the same input parameters (T)
18. C++ is a structured programming language supports Object-Oriented paradigm. (T)
19. The relation between the class Monkey and the class Animal is association relation (F)
20. We cannot create an object from a virtual base class.(F)
21. The destructor of a clas sis responsible for removing object from memory (F)
22. Multilevel inheritance not allowed in c++ (F)
23. Association is a special type of composition (F)
24. The term encapsulation refers to an object of a class that contains another object of another class inside it. (F)
25. When inherting from a base class, the derived class will inherit only the protected and public members of the base class (F)
26. Static member variables can only be modified through static member functions (F)
27. If a certain function is mad friend for class A, then that function can acess only the private member of class A. (F)
28. Compostion is special type of inheritance; composition is stronger relation between 2 objects than inheritance (F)
29. The this pointer is an array of pointers to all the objects created from a class (F)
30. When overloading a certain function, it is obligatory that we specify a different number of pararmters for the new function (F)
31. If the class Y is a sub-class of the class X, then an object of the class Y will only inherit the protected member (F)
32. A public member of a class can not be directly accessed by its name in the main() function. (T)

33. A friend function is allowed to access the private members of a class, but not the protected members of the class.(F)
34. In order to access a static variable of a class we have to write the class name before the variable name (T)
35. The this pointer is a pointer that points to the object that the method has been called for (T)
36. The constructor of a class is a member function that returns void.(F)
37. Encapsulation means extending a class to increase more member functions and member variables in a derived class.(F)
38. When a certain value is assigned to a static member variable at some point of the program, then it is not possible to assign a different value to this variable at a further point in the program.(F)
39. When we overload a function, we declare another function with the same name and we must specify a different number of parameters.(F)

40. Any static member can access all static and instance members. (F)
41. You can have more than a constructor in the same class. (T)
42. A destructor is a member function that removes object from memory. (F)
43. An abstract class is a class that contains any virtual functions. (F)
44. You can use more than a template for one class. (T)
45. A constructor is always declared void because it doesn't return a value. (F)
46. Aggregation and association are types of relationships between a base class and derived classes. (F)
47. The embedded object is an object declared with default constructor. (F)
48. You cannot overload the scope operator (::). (T)
49. You cannot overload the square brackets operator ([]). (F)
50. You can define a copy constructor in class. (T)

MCQ

1. When the compiler cannot differentiate between two overloaded functions, it is called
   a. Overloaded.
   b. Destructed.
   c. Overridden.
   d. Ambiguous

2. The Instance (non-static) variable
   a. It is a variable declared in the private section and can be accessed by any method in the same class
   b. It is a variable declared in any section in a class and can be accessed by any method in the same class.
   c. It is a variable declared inside any function and can be accessed by any other methods in the same class
   d. It is a variable declared inside any function and cannot be accessed by any other methods in the same class

3. If you design a class that needs special initialization tasks, you will have to create a(n):
   a. Setter method.
   b. Initializer Method.
   c. Constructor.
   d. Destructo

4. Which of the following statements is correct related to Object-Oriented concepts?
   a. Stand-alone function is one of the main parts of any Object-Oriented program.
   b. Every object is instantiated from a particular type.
   c. It's not necessary to make inheritance among classes.
   d. All objects from the same concrete class may perform different actions

   if only 1 choice. I probably will choose b.

5. The best description for the association relation among the classes
   a. When a class using object(s) from other class(es).
   b. Objects that are declared using the default constructor.
   c. An object from a class that is inherited from another class
   d. b and c

6. For a derived class from a base class, it can directly access:
   a. The private and protected members of the base class.
   b. Only the private members of the base class.
   c. Only the protected members of the base class.
   d. The protected and public members of the base class

7. Which of the following refers to "Hiding data behind class methods?
   a. Abstraction.
   b. Encapsulation
   c. Polymorphism
   d. Inheritance

   I think Encapsulation because we didn't take Abstraction

8. consider the following code:

```
class Base
{
public:
     int x;
     Base(int y)
     {
          x=y;
          ++x;
     }
};

class child : public Base
{
public:
     child(int c)
     {
          x+=1;
     }
     void print()
     {
          cout<<”x=”<<x;
     }
};

void main()
{
     child c(1);
     c.print();
}
```

a. x=1
b. x=2
c. x=3
d. x=4
e. Compilation Error.

9. **What will be the output when you compile and run the following piece of code?**

```
int sum(int a , int b) { return a+b;}
float sum(float a , float b) { return a+b;}
void main(void)
{
      clrscr();
      cout<<"sum ="<<sum(5,9)<<"\t" ;
      cout<<"sum ="<<sum(4.3,2.7) ;
 }
```

(2 Points)

- (●) a. sum = 14 sum = 7.0
- ( ) b. Compiler Error: add( ) function should be overloaded
- ( ) c. Compiler Error: Ambiguity between 'sum (int , int)' and 'sum (float, float)'
- ( ) d. sum = 14 sum = 6

10. What will be the output when you compile the following piece of code?

```
class Test
{
        int x ;
   public:
        float z ;
        int y ;
};
void main()
{
        Test t1 ;
        t1.x = 100 ;
        t1.y = 500 ;
        t1.z = 250.43 ;
 }
```

(2 Points)

- (●) a. Compiler Error regarding the variable x.
- ( ) b. Compiler Error regarding the variable y.
- ( ) c. Compiler Error regarding the variable z.
- ( ) d. The code will compile successfully.
- ( ) e. a and b.
- ( ) f. a, b, and c

11. What will be the output when you compile and run the following piece of code?

```
class Test
{
        public:
            virtual  void say() { cout<<"Test\t"; }
};
class NewTest : public Test
{
        public:
            void say() { cout<<"NewTest\t"; }
};
void main()
  {
        Test *pb ;
        Test myParent ;
        NewTest myChild ;

        pb = & myParent ;
        pb->say() ;
        pb = & myChild ;
        pb->say() ;
    }
```

(2 Points)

- (●) a. Test NewTest
- ( ) b. Test Test
- ( ) c. NewTest NewTest
- ( ) d. NewTest Test

12. **What will be the output when you compile and run the following piece of code?**

```
class Base
{
        public:
            Base() { cout<<"Welcome"<<"\t" ; }
 };
class Derived : public Base
{
        public:
            Derived() { cout<<"Hello"<<"\t" ; }
};
 void main()
 {
    Derived myD ;
 }
```

(2 Points)

- a. Hello
- b. Hello Welcome
- c. Welcome Hello
- d. Welcome

13. **What does the following piece of code do?**

```
void main()
{
   float *ptr ;
   ptr = new float(15);
}
```

(2 Points)

- a. Allocate space for 15 float contiguous variables that are initialized by 0.
- b. Allocate space for 15 float contiguous variables that are not initialized
- c. Allocate space for a float variable that is initialized by the value 15.
- d. Allocate space for a float variable that is not initialized.

14. In any abstract class, it has to include:(2 Points)

- a. At least a member function.
- b. At least a virtual function
- c. At least a static function.
- d. At least a pure virtual function
- e. All of the above

The body of the destructor of the base class is executed automatically:(2 Points)

15.

- (●) a. When any member function of the derived class is called.
- ( ) b. After the body of the destructor of the derived class is executed.
- ( ) c. Before the body of the destructor of the derived class is executed.
- ( ) d. We cannot determine exactly when it will be executed.

16.

'A car is a transportation mean that has a Motor, color and 4 wheels'.
'A bicycle is a transportation mean that has a handlebar, color and 2 wheels.

Which of the following best describes the previous statements as a set of classes? (2 Points)

- (●) a. 2 classes: A car class that has two attributes: color, motor and numberOfWheels, and a bicycle class that also has two attributes: color, handlebar and numberOfWheels.
- ( ) b. 3 classes: A Transportation_Mean class that has one attribute: color. A car class that inherits from the Transportation_Mean class and has two attributes: motor and numberOfWheels. And a bicycle class that also inherits from the Transportation_Mean class and has two attributes: handlebar and numberOfWheels.
- ( ) c. 1 class: A Transportation_Mean class that has an attribute for the: typeOfTransport.
- ( ) d. 3 classes: A Transportation_Mean class that has one attribute: numberOfWheels. A car class that inherits from the Transportation_Mean class and has two attributes: motor and color. And a bicycle class that inherits from the car class and has one attribute: handlebar.

17.

'A plane is a machine that has a motor and has wings'.
'A refrigerator is a machine that has a motor and has shelves'.
Which of the following best describes the previous statements as a set of classes?

a) 3 classes: A **machine** class that has one attribute: *motor*. A **plane** class that inherits from the **machine** class. And a **refrigerator** class that inherits from the **plane** class.

b) 3 classes: A machine class that has one attribute: motor. A plane class that inherits from the machine class. And a refrigerator class that also inherits from the machine class. →

c) 2 classes: A plane class that has two attributes, and a refrigerator class that also has two attributes. →

d) 1 class: A **machine** class that has an attribute for the *type of machine*.

18.

The following are examples of Object-Oriented advantages: (choose all that applies)(2 Points)

- [ ] a. Simplicity
- [x] b. Reusability
- [x] c. Encapsulation
- [x] d. Modularity
- [x] e. Abstraction

I. The Embedded object is:(2 Points)

- ( ) a. An object declared with default constructor.

19. I. The Embedded object is:(2 Points)

- a. An object declared with default constructor.
- b. An object has function overloading.
- c. An object from a class inherited from another class.
- d. An object included by another object.

20. 33. **What will be the output when you compile and run the following piece of code?**

```
class Test
{
        int x;
    public:
        Test()
        {
            x = 0;
        }
        Test(int y)
        {
            x = y++;
        }
        Test(Test &r)
        {
            x = r.x++;
        }
        void print()
        {
            cout<<x ;
        }
};
void main()
{
    Test t(1) ;
    t.print() ;
    Test x(t) ;
    x.print() ;
    t.print() ;
}
```

(2 Points)

- 121
- 112
- 222
- 122

21. 2. Given the class below, what changes do we need to make to the class in order to have a facility to add the values of an object's data to another?

```
class Test
{
        int x ;
        int y ;
    public:
        Test( )
        {
            x = 0 ;
            y = 0 ;
        }
        void setXY(int a , int b)
        {
            x = a ;
            y = b ;
        }
 };
```

(2 Points)

- a. Do operator + overloading.
- b. Declare a third temporary member integer.
- c. Create a new member function: void Add( ), that takes no parameters.
- d. All of the above.

22. Object-Oriented is used because: (2 answers)

- The code is smaller.
- The program is faster.
- The code is reusable.
- The code is easy to maintain.

23. We use the inheritance for:

- Increasing more member functions only to the base class.
- Extending base class functionality.
- Including the base class.

24. The following are examples of polymorphism: (2 answers)

- Operator overloading
- Function overriding
- Friend functions
- Constructor overloading.

25.

```
class XX
{
        int i;
        char c;
        void seti (int r) { }
        char getc ( ) { return c; }
};
void main( )
{
        XX x;
        x.seti(8);
}
```

On running, the previous code:

- compiles with no errors
- the compiler generates an error because the class XX cannot be instantiated because it has no public constructor.
- The linker generates an error because the class XX cannot be instantiated because it has no public constructor.
- The compiler generates an error because the method "seti" is non accessible through object x.

26.

```
void printNumber( )
{
        Int num=5;
        for (i=0 ; i<10 ; i++)
        { cout<<num; }
        int i;
        char c;
}
```

In C++, on calling the previous function:

- An error occurs because i & c cannot be declared after the loop (invalid declaration).
- An error occurs because the var i is not visible in the for loop (undefined symbol i)
- No error occurs.

27. The constructor of the base class called automatically:

- When any member function of the derived class called.
- After the constructor of the derived class started.
- Before the constructor of the derived class started.
- We do not know when exactly it will be started.

28.
```
class X
{
        int k;
        public:
                void setK(int k) {........}
};
```
Which can best be written in the space?

- this->k=k;
- k=k;
- this.k=k;

29.
```
class Person
{
        public:
                virtual void talk( )==0;
};
class Adult : public Person
{
        public:
                void talk() { cout<<"Hello"; }
};
class Female : public Person
{
        public:
                void talk( ) { cout<<"Hi"; }
};
void main( )
{
        Person p;
        Adult a;
        Female f;
}
```
The previous code:

- Compiles successfully.
- Produces an error. -- Person is abstract class

30.

```
class Child : public Base
{
        public:
                Child(int x) { }
                Child(int x,int y) : Base(x,y)
                { }
};
```

What are the expected constructors that must exist in the base class?

- Base( ), Base(int,int)
- Base( ), Base(int)
- Base(int), Base(int,int)
- Base(int,int)

31. The class that contains a pure virtual function is called:

- Base class.
- Derived class.
- Abstract class.
- Embedded class.

32.

```
class XX
{
        int i;
        char c;

        public:
                void seti(int r) { }
                void getc( ) { return c; }
                void printHi( ) { cout<<"Hi"; }
};
void printHi( ) { cout<<"Hello"; }
void main( )
{
        XX x;
        x.printHi( );
}
```

The previous code prints:

- Hello.
- Hi.
- Hi then Hello
- None of the above, it generates an error. Void getc can't return values

33.

```
. class Student
{
        char name[20];
        public:
                Student( ) { strcpy(name, "Anonymous"); cout<<name; }
                Student(char* nm) { strcpy(name,nm); cout<<name; }
                ~Student( ) { }
};
void main( )
{
        Student s1(Mona);           ......1
        Student *s2;                ......2
        s2 = new Student("Ali");    ......3
        delete s1;                  ......4
        delete s2;                  ......5
}
```

In the previous code:

- It compiles successfully.
- Line 4 generates an error.
- Line 5 generates an error.
- Both lines 4,5 generate errors.

34.

```
. class Test
{
        public:
                void print(int i) ..1
                { cout<<"int version"; } .. 2
                void print(char *c) .. 3
                { cout<<"char version"; } ..4
};

void main( )
{
        Test t; ..5
        char ch='p'; ..6
        t.print(ch); ..7
}
```

Which of the statements below is true?

- Line 4 will not compile, because void methods cannot be overloaded.
- Line 7 will not compile, because there is no version of print that takes a char.
- The code will compile and produce the following output: int version.
- The code will compile and produce the following output: char version.

35.

```
class Base
{
        public:
                int x;
};

class Child : private Base
{
        public:
                int y;
};

void main( )
{
        Child c;
        c.x=10;
        cout<<"x="<<x;
}
```

What happens in program?

- Compiler error because x is not declared in Child class.
- Compiler error because of the Child inheritance from private class.
- x=10
- Runtime error.

36.

A constructor is:

a) A member function that removes an object from the memory.

b) A member function that is called automatically when an object is being removed from the memory.

c) A member function that is used to initialize variables of a declared object. →

d) None of the above.

37. **Given the class below, what changes do we need to make to the class in order to have a facility to test if the values of two objects' data are equal?**

```
class MyClass
{
   private:
           float a ;
           float b ;
   public:
         MyClass( )

               a = 0 ;
               b = 0 ;
         }

         void setValues(int x , int y)
         {
               a = x ;
               b = y ;
         }
};
```

a) Declare a third temporary member integer.
b) Do operator = = overloading. →
c) Create a new member function: `void testEqual( )`, that takes no parameters.
d) None of the above.

38. **Which of the following statements are true about destructors?**
a) Destructors are special functions with the same name of the class and are preceded by a tilde character (~).
b) Destructors cannot be overloaded.
c) A destructor has no return type.
d) All of the above. →
e) None of the above.

39. **Which of the following most closely describes the process of overloading?**
a) A class with the same name replaces the functionality of a class defined earlier in the hierarchy.
b) A method with the same name completely replaces the functionality of a method defined earlier in the hierarchy.
c) A method with the same name but different parameters gives multiple uses for the same method name. →
d) A class is prevented from accessing methods in its immediate ancestor.

40. **What will happen when you attempt to compile the code below?**

```
class MyClass
{
        int x ;

   public:
        int y ;
};

void main()
{
   MyClass obj ;
   obj.x = 100 ;
   obj.y = 500 ;
}
```

a) Compiler Error because x is not accessible. →
b) Compiler Error because y is not accessible.
c) Compiler Error because **MyClass** is an abstract class and can not be instantiated.
d) a and **b**.
e) The code will compile successfully.

41. **The following function prototype uses the default arguments feature:**

```
void myFunc(int x=3 , int y) ;
```

is the function valid this way?

a) Yes, it is ok to give a default parameter for x and not for y.
b) No, a function must always declare all its variables with default values.
c) No, you must only give a default parameter for y but not for x.
d) No, there should not be any arguments without default values to the right of the default arguments. →

42. **A derived(sub)(child)(inherited) class can directly access:**

a) The protected and public members of the base class. →
b) The private and protected members of the base class.
c) Only the protected members of the base class.
d) Only the private members of the base class.
e) The public, private, and protected members of the base class.

43. **Assume you have a class X that contains an object of class Y. Assume that we declare an object of class X in the main( ) function. When will the body of the constructor of class Y be executed?**

a) When any member function of the class X is called.
b) After the body of the constructor of class X is executed.
c) Before the body of the constructor of class X is executed. →
d) We can not determine exactly when it will be executed.

44. **An embedded (aggregated) object is:**

a) An object that s declared with the default constructor.

b) An object that is included by another object. →

c) An object from a class that is inherited from another class.

d) b and **c**.

e) None of the above.

45.

```
class Point
    {
     float x, y;
    Point (float a, float b) { x=a; y=b;}
    Point () { x=0; y=0; }
  }
```

To write copy constructor to class point, what would be its signature? (3 Points)

- ○ Point (Point)
- ● Point (Point &)
- ○ Point & Point (Point &).
- ○ Point & Point (Point)

46.

```
class Nice
    {   int a ;
        public:
            Nice( ) { a = 0 ; }
            Nice(Nice & myN)
                {   this -> a =  myN.a ;
                    cout<<"I am the copy constructor  " ;
                }
            void setA(int m)  { a = m; }
             int getA()   { return a ; }
    };
void show(Nice &obj)
{
cout<<"I am the show function, value is: " << obj.getA() ;
}
void main()
{   Nice n1;
    n1.setA(15) ;
    show(n1) ;
}
```

(3 Points)

- (●) I am the show function, value is: 15.

47.

If we did not specify a constructor to the class, then :

(3 Points)

- ( ) we won't be able to create object of class
- ( ) we won't be able to create object of class, and compiler will give compilation error
- ( ) we won't be able to create object of class, and compiler will give warning
- ( ) it will generate run-time error
- (●) None of the above

48.

We can overload Destructor in the class

(1 Point)

- ( ) True
- (●) False

Which of the following statements are true about constructor?
(3 Points)

- [x] A constructor can be overloaded.

49. - [x] A constructor is a special member function with the same name of the class.

- [ ] A constructor can return a primitive or an object reference.

- [ ] All the above

What will be the output when you compile and run the following piece of code?

important

50.
```
class Parent
  {
     protected:
              int x;
     public:
              Parent(int m)
                      { x = m ; }
      friend void display( );
   };
class Child : public Parent
  {
     private:
              int y;
      public:
            Child(int m, int n) : Parent(m)
             {  y = n ;  }
   };
  void display ()
            {
              Child c(3,4);
              cout <<"x="<<c.x<<"y="<<c.y; // Line 1
            }

   void main ()
            {
              display();
            }
```

(3 Points)

- ( ) Compilation Error at Line 1, Child::x is inaccessible
- ( ) Compilation Error at Line 1, Child::y is inaccessible
- (•) A and B
- ( ) The code compiles successfully.

51.

```
class Super
{
    protected:
                Super(int a)
                 {
                      this.a = a;     // Line 1
                 }
   private:                          // Line 2
              int a;
};
class Sub: public  Super
{
     public:
              Sub(int a):Super(a)  { }
     public:
             Sub()              //Line 3
              {
              this.a= 5;        //Line 4
             }
};
```

Which steps will allow Sub to compile?
(3 Points)

- [ ] Class Sub compile successfully.
- [ ] Comment Line 2
- [ ] In Line 1 and 4 to, use (*this).a instead of this.a;
- [ ] Change Line 1 and 4 to, this(a);
- [ ] Change Line 3 to, Sub(): Super(5)
- [ ] Change Line 3 to, Sub(): this(5)
- [ ] All the above.

52. What will be the output when you compile and run the following piece of code?

```
class Base
        {       public:
                    Base() { cout<<"Welcome     " ; }
        };
class Derived : public Base
        { public:
                Derived() { cout<<"Hello     " ; }
        };
void main()
      {  Base myBase ;
         Derived myDerived ;
      }
```

(3 Points)

- ○ Welcome Hello
- ○ Hello Welcome
- ○ Welcome Hello Welcome
- ◉ Welcome Welcome Hello

53. In order to turn a class into an abstract class, which of the following do we need to do?

(3 Points)

- ○ Write the abstract keyword before the name of the class.
- ○ Make the class a pure virtual class.
- ◉ Write one or more pure virtual functions inside the class.
- ○ A and C.
- ○ None of the above

54. . A protected member of a class can be directly accessed by its name inside another class if and only if that other class is a child of that class.

(1 Point)

- ◉ True
- ○ False

). Which of the following most closely describes the process of overriding?
(3 Points)

- ○ A class with the same name replaces the functionality of a class defined earlier in the hierarchy.
- 55. ● A function with the same name replaces the functionality of a function defined earlier in the inheritance hierarchy.
- ○ A function with the same name but different parameters gives multiple uses for the same function name.
- ○ Making a class abstract so that no objects can be declared from it.

Assume you have a class M that contains a pointer to an object of class N. Assume that we declare an object of M in the main( ) function. When will the body of the constructor of class N be executed?
(3 Points)

56.
- ● When any member function of the class M is called.
- ○ After the body of the constructor of class M is executed.
- ○ Before the body of the constructor of class M is executed.
- ○ None of the above.

57.

```
. class Parent
        { public:
                int x;
                Parent(int m)   { x = m ; }
        };
    class Child : protected Parent
        {
            public:
                int y;
                Child(int m, int n) : Parent(m) {   y = n ;  }
        };
     class GrandChild : public Child
        {    int z ;
            public:
                GrandChild(int a, int b, int c) : Child(a,b) {  z = c ; }
        };
    void main( )
        {  GrandChild obj(3,5,7);
            cout<<"Value of x is: "<<obj.x <<endl ;          //Line 1
            cout<<"Value of y is: "<<obj.y <<endl ;          //Line 2
            cout<<"Value of z is: "<<obj.z <<endl ;          //Line 3
        }
```

(3 Points)

- [x] Compiler Error at Line 1
- [ ] Compiler Error at Line 2
- [x] Compiler Error at Line 3
- [ ] The code compiles successfully.

58. 22. What will be the output when you compile and run the following piece of code?

```
class Tester
 {   public:
                int x ;
               static int var ;
                Tester(int a) {   x = a ; }
               static void myFunction(int a)
                   {
                   Tester obj(9) ;      // Line 1
                   obj.x=a ;             // Line 2
                      cout<<obj.x;
                   }
  };
  int Tester::var=0;
 void main()
 {
   Tester myT(7) ;
   Tester::myFunction(15);
   cout<<myT.x ;
   cout<<Tester::var<<endl ;       // Line 3
  }
```

(3 Points)

- (●) 15 7 0
- ( ) 7 15 0
- ( ) 15 9 0
- ( ) 9 15 0
- ( ) Compilation error at line 1

59. . The term " Composition " refers to an object of a class that contains another object
of another class inside it.

(1 Point)

- (●) True
- ( ) False

60. Which of the following is true about an object member function?
(4 Points)

- [ ] It can be called using the name of the class.
- [x] It can access static variables of the class.
- [x] It has a "this" pointer as an implicit parameter passed to it.
- [ ] It can access the instance variables.
- [ ] It cannot be overloaded.
- [ ] It can call other member functions from inside it.

61. 12

Which of the following is true about an object member function?
(2 Points)

- ( ) A) It can be called using the name of the class
- ( ) B) It can access static variables of the class
- ( ) C) It has a "this" pointer as an implicit parameter passed to it
- ( ) D) It can access the instance variables
- ( ) E) It cannot be overloaded
- ( ) F) It can call other member functions from inside it
- ( ) A & B & C
- ( ) B & C & E
- ( ) A & C & D & F
- ( ) B & C & D & F

62. Which of the following is true about the function prototype below?
void myFunc (int myDef=17, int myVar , int myNormalVar=5) ;
(3 Points)

- (•) We should also give a default value to myVar.
- ( ) We must only give a default parameter for myNormalVar and not the others.
- ( ) The function is correct in that way.

27. What does the following piece of code do?

```
void main( )
{
 float *ptr ;
  ptr = new float[15];
}
```

(3 Points)

63.

- ○ Allocate space for a float variable that is not initialized
- ○ Allocate space for an array of 15 float elements that are not initialized
- ○ Allocate space for an array of 15 float elements that is initialized by the value 0
- ○ Allocate space for an array of 15 float elements where all the elements are initialized by the value 15
- ◉ Compiler Error.

. Static member variable can only be modified through static member functions.
(1 Point)

64.

- ○ True
- ◉ False

. class Stack

```
    {
      int tos,size;
      int * st;
      Stack( int s=5) { tos=0;size=s; st=new int[size];}
      ~ Stack( ){ delete []st;}
    };
void main()
{
  Stack s1(20);
}
```

65. what shall we add to the above class Stack to declare another object s2 from class Stack where s2 is declared in terms of s1; Stack s2(s1)?
(3 Points)

- ○ We must specify overload of assignment operator for class Stack.
- ○ We must define a copy constructor to class Stack
- ◉ A and B
- ○ This situation cannot be achieved in C++, however, it has been solved in other programming languages.

specify the way of declared Stack s2(s1) // only call cpy ctor

66. What will be the output when you compile and run the following piece of code?

```
class Parent
 {
     int x;
     Parent(int m){ x = m ; }
 };
class Child : public Parent
 { public:
         int y;
         Child(int m, int n) : Parent(m) //Line 1
        {  y = n ;  }
 };

void main( )
{
   Child obj(3,5,7);          //Line 2
  cout<<"Value of x is: "<<obj.x <<endl ;  //Line 3
  cout<<"Value of y is: "<<obj.y <<endl ;  //Line 4
}
```

(3 Points)

- [x] Compiler Error at Line 1
- [x] Compiler Error at Line 2
- [x] Compiler Error at Line 3

67. What will be the output when you compile and run the following piece of code?

```
class Parent
        {
          protected:
             int myVar;
          public:
             Parent(int x) { myVar=x; }
             void powerTwo(){ cout<<myVar*myVar; }
             virtual void powerThree() { cout <<myVar*myVar*myVar; }
        };
    class Child:public Parent
        {
          protected:
               int myData;
          public:
               Child(int a, int b) : Parent(a) {myData= b;}
             void powerTwo() { cout <<myData*myData; }
             void powerThree() { cout<<myData*myData*myData; }
        };
    void main()
    {
       Child myCh(2,3);
       Parent *myPtr;
      myPtr = &myCh;
      myPtr->powerTwo(); //Line1
      myPtr->powerThree(); //Line 2
}
```

(3 Points)

- ○ 4 8
- ◉ 4 27
- ○ 9 27
- ○ 9 8
- ○ Compilation Error at Line1

What will be the output when you compile and run the following piece of code?

68.

```
class Parent
        {  int y;
           static int z;
           public:
                    Parent( )
                        {
                          z=0;   // Line1
                        }
                    Parent (int a=5)   //Line 2
                        {
                          y=a;
                        }
        };
void main( )
        {
           Parent d(4);     //Line 3
           Parent m;        //Line 4
        }
```

(3 Points)

- ○ Compilation Error at Line 1, an object member function cannot access a static member
- ○ Compilation Error at Line 2, constructor should initialize static member (z=0;)
- ○ Compilation Error at Line 3
- ◉ Compilation Error at Line 4.
- ○ The code compiles successfully.