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

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

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)

- [x] 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)

- [x] 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. 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.

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)

a. When any member function of the derived class is called.

15. 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.

. ***'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.

16. 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.

'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.

17. 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*.

The following are examples of Object-Oriented advantages: (choose all that applies)(2 Points)

- [ ] a. Simplicity
- [x] b. Reusability

18.

- [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.

```
class XX
{
        int i;
        char c;
        void seti (int r) { }
        char getc ( ) { return c; }
};
void main( )
{
        XX x;
25.     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.

```
void printNumber( )
{
        Int num=5;
        for (i=0 ; i<10 ; i++)
        { cout<<num; }
        int i;
        char c;
26. }
```

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

```
. class Student
{
        char name[20];
        public:
                Student( ) { strcpy(name, “Anonymous”); cout<<name; }
                Student(char* nm) { strcpy(name,nm); cout<<name; }
                ~Student( ) { }
};
void main( )
{
33.     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.

```
. class Test
{
        public:
                void print(int i) ..1
                { cout<<”int version”; } .. 2
                void print(char *c) .. 3
                { cout<<”char version”; } ..4
};

void main( )
{
34.
        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.

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 (~).
38. **b)** Destructors cannot be overloaded.
c) A destructor has no return type.
d) All of the above. →
e) None of the above.

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.
39. **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.

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

40.
```
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.

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?

43. 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)

- (selected) 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
- (selected) None of the above

48. We can overload Destructor in the class

(1 Point)

- ○ True
- (selected) False

49. 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

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

important

```
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)

- [x] 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)

- [x] 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.

63\. 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.

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

- ◯ True
- ◉ False

65\.

```
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.

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

68. 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.