- (●) False

5. virtual inheritance let the child insure that there is only one instance of the parent class (1 Point)

- (●) True
- ( ) False

6. We must overload the = operator when implementing the copy constructor (1 Point)

- ( ) True
- (●) False

7. cout is a member function of the built-in ostream class where the operator<< is overloaded (1 Point)

- (●) True
- ( ) False

8. 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

9. 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 << "Sum of all variables is: "<<x + y +z;}
};
void main()
{
    Base *myPtr;
    Derived myDerived(15, 5,4);
    myDerived.showSum();
}
```

The above code:

(2 Points)

- ○ 24
- ○ 7
- ◉ Compilation Error
- ○ None of the above

10. consider the following code:

```
class Base
{
    public:
        int x;
        Base(int y)
        {
                x=y;
                ++x;
```

```
        }
};
class child : public Base
{
    public:
        child(int c):Base(c)
        { x=1; }
        void print()
        { cout<<"x="<<x;  }
};
void main()
{
        child c(1);
        c.print();
}
```

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

- ○ x=3
- ◉ x=1
- ○ x=4
- ○ x=2

11. 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)

- ○ 121
- ◉ 122
- ○ 222
- ○ 233

12. 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 : public BB, public CC
{
    public:
        XX(int m, int n, int p) : BB(m,n), CC(n, p)
        {}
};
XX myObject(3, 4, 5);
```

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

- ○ a = 4, b = 3, c =4
- ○ a = 5, b = 3, c =4
- ◉ Undetermined Values
- ○ Compliation Error

13.
```
class MyClass
{               int x, y;
                static int count;
        public:
                MyClass(int m)
                {
                        x = y = m;
                        count++;
                }
                ~MyClass()
                {
                        count--;
                }
                static int GetCount()
                {
                        return count;
                }
          friend MyClass FillMyClass();
          friend void PrintMyClass(MyClass obj);
};
int MyClass::count = 0;
```

```
MyClass FillMyClass()
{
   MyClass obj(7);
   obj.x = 5;
   obj.y = 7;
   return obj;
}
void PrintMyClass(MyClass obj)
{
   cout << "X = " << obj.x ;
   cout << "Y = " << obj.y ;
}
```

What is the Output of:

```
int main()
{
    MyClass obj(8);
    obj = FillMyClass();
    PrintMyClass(obj);
    cout<<obj.GetCount();
    return 0
}
```

(2 Points)

- ○ X=5 Y=7 0
- ◉ X=5 Y=7 1
- ○ X=5 Y=7 -1
- ○ Other

14. Consider the following code

```
class Base
{
    protected:
            int x, y;
    public:
            Base() {x = y = 3;}
            int GetX(){return x;}
            int Sum() {return (x+y);}
            virtual int Product(){return (x*y);}
```

```
};
class Derived
{
        int z;
    public:
        Derived() {z = 5;}
        virtual int Sum() {return (x+y+z)};
        int Product(){return (x*y*z);}
};
int main()
{
    Base *ptr;
    ptr = new Derived();
    cout<< ptr->Sum()
}
```

The previous code output is: (2 Points)

- ○ 6
- ○ 11
- ○ Unexpected Value
- ◉ Complier error

15. Consider the following class:

```
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 << endl; }
```

```
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
- ○ 10
- ○ 20 10
- ◉ None of the above

16. Consider the following class:

```
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)

- ○ DerivedOne
- ◉ DerivedTwp
- ○ Error
- ○ None of the mentioned

17. WE can write a class object in a text file by: (2 Points)

- ○ a) Override the << operator in ofstream

- [x] b) Override the << operator in ostream
- [ ] c) Both a and b
- [ ] d) Either a or b

18. 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;
b = g(c);
cout << A::m_copy_ctor_calls;
A& d = f(c);
cout << A::m_copy_ctor_calls <<endl;
return 0;
} (2 Points)

- [ ] 245
- [x] 254
- [ ] compile error

- ◯ None of the above

19.
```
class MyClass
{
   int x;
   public:
     MyClass()
     { x = 5; }
     friend ostream& operator <<(ostream& o, MyClass m);

};
ostream& operator <<(ostream& o, MyClass m)
{
   o<<m.x<<endl;
   return o;
}

int main()
{
  MyClass obj;
  ofstream out("C:\\test\\myfile.txt",ios::out);
  out << obj;
  out.close();

  return 0;
}
```
(2 Points)

- ◯ Write 5 to myfile.txt
- ◯ error: << operator is not overloaded for ofstram class
- ◉ will compile without error, but don't write to myfile.txt
- ◯ Compile, Create myfile.txt but

20.
```
class MyClass
{
   int x;
```

```
    public:
      MyClass()
      { x = 5; }
      int GetX(){return x;}
      friend istream& operator >>(istream& i, MyClass* m);

};
istream& operator >>(istream& i, MyClass* m)
{
   i>>m->x;
   return cin;
}
int main()
{
  MyClass obj;
  cin>>&obj;
  cout<<obj.GetX();
  return 0;
}
```

(2 Points)

- ○ Compile time error
- ○ run time error
- ◉ fill the MyClass object
- ○ Nothing

21.
```
class MyClass
{
  public:
  MyClass()
  {cout << "Call of Default Constructor"<<endl;}
  MyClass(int m)
  {cout << "Call of Parametrized Constructor"<<endl;}
  MyClass(MyClass &m)
  {cout << "Call of Copy Constructor"<<endl;}
  ~MyClass()
  {cout << "Call of Destructor"<<endl;}
  MyClass operator= (MyClass obj)
  {cout << "Call of Overloading of = operator"<<endl;
```

```
    return obj;}
};
MyClass FillClass()
{  MyClass obj;
   cout << "Call of Fill method"<<endl;
   return obj;
}
void PrintClass(MyClass obj)
{  cout<< "Call of Print method"<<endl;}
int main()
{
 MyClass obj;
 obj = FillClass();
 PrintClass(obj);
 return 0;
}
```

What are the number of output statement (2 Points)

- (●) 10
- ( ) 11
- ( ) 12
- ( ) 13

22.
```
class MyClass
{
  static int Count;
  public:
    MyClass()
    {Count++;}
    ~MyClass()
    {Count--;}
    static int GetCount()
    {return Count;}
};
int MyClass::Count = 0;
void Print(MyClass obj)
{}
int main()
```

```
{
  MyClass obj[5];
  for(int i = 0 ; i < 5 ; i+=2)
  {
     Print(obj[i]);
  }
  cout<<MyClass::GetCount();
  return 0;
}
```

The Output of the previous code: (2 Points)

- (●) 0
- ( ) 1
- ( ) 2
- ( ) 3

23.
```
class Base
{
   protected:
     int x;
   public:
     Base(int a)
     { x = a; }
     Base(Base& rb)
     { x = rb.x; }
     void SetX(int a)
     { x = a; }
     int GetX()
     { return x; }
};
class Derived : public Base
{
     int y;
   public:
     Derived()
     { y = 0; }
     Derived(int a, int b) : Base(a)
     { y = b; }
```

```
    Derived(Derived& rd)
    { y = rd.y; }
    void SetY(int b)
    { y = b; }
    int GetY()
    { return y; }
    int Product()
    { return (x * y); }
};

int main()
{
    Derived obj(3, 4);
    Derived drv(obj);
    cout<< drv.Product() << endl;
    return 0;
}
```

(2 Points)

- 12
- 4
- Compile Error
- None of the above

24.

```
class Base
{
  protected:
    int x;
  public:
    Base()
    { x = 0; }
    Base(int a)
    { x = a; }
    Base(Base& rb)
    { x = rb.x; }
    void SetX(int a)
    { x = a; }
    int GetX()
    { return x; }
```

```
};
class Derived : public Base
{
      int y;
   public:
      Derived()
      { y = 0; }
      Derived(int a, int b) : Base(a)
      { y = b; }
      Derived(Derived& rd)
      { y = rd.y; }
      void SetY(int b)
      { y = b; }
      int GetY()
      { return y; }
      int Product()
      { return (x * y); }
};

int main()
{
    Derived obj(3, 4);
    Derived drv(obj);
    cout<< drv.Product() << endl;
    return 0;
}*/
```

(2 Points)

- ◯ 0
- ◯ 4
- ◯ 12
- ◯ Compile error

25. Multiple inheritance from 2 parents that inherit the same grand parent always perform ambiguity error (1 Point)

- ◉ True

- ○ False

26. Inline method can't be applied for non member method (1 Point)

- ◉ True
- ○ False

27. Call by reference give the same effect of call by address, as the are both access the original data through its address (1 Point)

- ◉ True
- ○ False

28. When the default constructor is absent, the compiler always create an auto default constructor (1 Point)

- ◉ True
- ○ False

29. cin is a variable from istream class defined in iostream (1 Point)

- ◉ True
- ○ False

30. As Non static method is called by an object from the class, so it can deal with non static member only (1 Point)

- ○ True
- ◉ False