Qus1. What is class and object? and how many access specifier in class?

Ans:- Class:- is a user defined datatype which holds its data member and member function in another word we can say class is a collection of data member and member function. which can accessed and use by creating object of that class. A C++ class is like a blueprint for an object.

Object:- An object is an instance of class. whenever class is defined no memory is allocated but when when object is initialized memory is allocated of that class.

Qus2. What is Static member variable?

Ans 1. Declared inside the class body

2. also known as **class member variable**

3. **they must be defined outside the class otherwise show error.**

4. static member variable does not belong to any object, but to whole class.

5. there will be **only one copy** of static member variable for the whole class.

6. any object can use the same copy of class variable.

7. they can also be used with the class name.

```
class Account{
 private:
   int balance; //instance member variable
   static float roi; //static member variable or class variable and here it not get memory
 public:
   void setBalance(int b){
   balance = b}
   static void setRoi(int b){ //static member function
   roi= b}
};
float Account :: roi = 1.5; // now it get memory
int main(){
  clrscr();
```

```
  Account a1,a2; //it do not contain roi
  //Account :: roi = 4.5; // we cannot access because it is private so make setroi
  Account :: setRoi(4.5);
}
```

Qus2. What is friend function?

Ans:- is not a member function of a class to which it is a friend. //ye us class ka friend hai member fun nhi same as non memb f but can acces class member

- It is declared in the class with friend keyword.
- **it must be defined outiside** the class to which it is friend.
- it cannot access members of the class directly.
- **it has no caller object**.
- it **should not be defined with membership label(::)**
- ***Friend function can become friend to more than one class***

Example 1:-

```
class Complex{
   private:
      int a,b; //by default private
   public:
     void setData(int x,int y){
          a=x; b=y;
      }
      void showData(int x,int y){
          cout<<"a"<<a<<"b"<<b;
      }
      friend void fun();
};
void fun(Complex c){      // can not define  Complex::fun()
  cout << "sum is"<< c.a+c.b; // cannot access directly like cout << "sum is"<<a+b; becz malum nahi chalega ki ye c1 ka h ya c2 c3 so recieve obj
```

```
}
void main(){
Complex c1;
c1.setData();
fun(c1); //not like this c1.fun(); so we pass object
}
```

Example 2:

```
//Friend function can become friend to more than one class
class B;
class A{
  private:
    int a;
  public:
    void setData(int x){a=x;}
    friend void fun(A,B);
}
class B{
  private:
    int b;
  public:
     void setData(int x){b=x;}
    friend void fun(A,B);
}
void fun(A o1,B o2){
cout<<"sum is"<<01.a+o2.b;
}
```

```
void main(){
A obj1;
B obj2;
obj1.setData(3);
obj2.setData(2);
fun(obj1,obj2);//5
}
```

Example 3:Overloading of operators as a friend function

```
class Complex{
   private:
      int a,b; //by default private
   public:
     void setData(int x,int y){
          a=x; b=y;
      }
      void showData(int x,int y){
          cout<<"a"<<a<<"b"<<b;
      }
      friend Complex operator +(Complex , Complex);
};
Complex operator+(Complex X,Complex Y){
        Complex temp;
        temp.a = X.a + Y.a;
        temp.b = X.a + Y.b;
        return temp;
}
```

```
void main(){
complex c1,c2,c3;
c1.setData(3,4);
c2.setData(5,6);
c3 = c1+c2;  // c3 = operator+(c1,c2)   + 1 argument in friend function
c3.showData();
}
```

Example 4:Overloading of unary operators as a friend function

```
class Complex{
   private:
      int a,b; //by default private
   public:
     void setData(int x,int y){
          a=x; b=y;
      }
      void showData(int x,int y){
          cout<<"a"<<a<<"b"<<b;
      }
      friend Complex operator -(Complex);
};
Complex operator-(Complex X){
          Complex temp;
          temp.a = -X.a;
          temp.b= -X.b;
          return temp;
      }
```

```
void main(){
complex c1,c2;
c1.setData(3,4);
c2=-c1;    //c2=c1.operator-()
c2.showData();
}
```

Example 5:Overloading of insertion and extraction operators as a friend function

```
class Complex{
   private:
      int a,b; //by default private
   public:
     void setData(int x,int y){
          a=x; b=y;
      }
      void showData(int x,int y){
          cout<<"a"<<a<<"b"<<b;
      }
      friend  ostream& operator<<(ostream& , complex);
      friend  istream& operator>>(istream& , complex&);
};
ostream& operator<<(ostream &dout,Complex C){
 cout<<"a"<<Cq.a<<"b"<<C.b;
 return dout;
}
istream& operator>>(istream& din, complex& C){
 cin>>C.a>>C.b;
```

return din; // **return for achieve cascadding** (because we are using din like cin>>a>>b>>c; for add more)

}

void main(){

complex c1;

cout<<"Enter a complex number:";

cin>>c1; //give error or cin.operator>>(c1);

cout<<"you entered :";

cout<<c1<<c2; // operator(operator<<(cout,c1),c2); when as a friend function

}

Example 6: **Member function of one class can become friend to another class**

```
class A{
   public:
   void fun()
   {  }
   void fun2()
   {  }
};
class B{
  friend void A :: fun();
  friend void A :: fun2();
}
void fun(){
}
```

Example:-

```
#include <iostream>
```

```
using namespace std;

class A; // forward declaration of A needed by B

class B
{
    public:
        void display(A obj); //no body declared
};

class A
{
    int x;
    public:
        A()
        {
            x = 4;
        }
        friend void B::display(A);
};

void B::display(A obj)
{
    cout << obj.x << endl;
}

int main()
{
    A a;
    B b;
    b.display(a);
    return 0;
}
```

or

//instead of making friend of all member function of class A do class A full friend

```
class B{
  friend class A;
};
```

Example:-

```
class ABC;// forward declaration that we have ABC class
class XYZ {
   int x;
   public:
      void set_data(int a) { x=a; }
      friend void max(XYZ,ABC);
};
class ABC{
   int y;
   public:
      void set_data(int a) { y=a;  }
      friend void max(XYZ,ABC);
};
void max(XYZ t1,ABC t2){   //definition of friend function
   if(t1.x>t2.y)
      cout<<t1.x;
   else
      cout<<t2.y;
}

main()
```

```
{
    ABC _abc;
    XYZ _xyz;
    _xyz.set_data(20);
    _abc.set_data(35);
    max(_xyz,_abc);  //calling friend function
    return 0;
}
```