Objective:

- The purpose of this quiz is to focus on the very basic fundamental concepts learned so far in previous lectures.

Question No. 1: **(**10**)**

Consider the code below and answer the questions from Part-A to Part-E.

```
#include<iostream.h>
class Associate
{
private:
        int ix;
public:
        Associate(int n=0)
        {
                ix = n;
        }
        void show() const
        {
                cout<<"Hmm, Value is
"<<ix<<endl;
        }
};
class Association
{
private:
        Associate* pa;
public:
        Association()
        {
                pa=0;
        }
        void addAssociate(Associate& other)
        {
                pa=&other;
        }
        void addAssociate(Associate* pother)
        {
                pa=pother;
        }
        void removeAssociate()
        {
                delete pa;
                pa=0;
        }
void showAssociate() const
  {
        cout<<"object address is: "<<
        pa<<endl;
        pa->show();
  }
  Association& operator=
                ( const Association & other)
  {
        pa = other.pa;
        return *this;
  }
  ~Association()
  { }
};

 void main()
{
  Association a;
  Association b;
  Associate x(17);
  Associate* pa = new Associate;
  a.addAssociate(*pa);          //Line 1
  pa->show();                   //Line 2
  pa=0;                         //Line 3
  a.addAssociate(x);            //Line 4
  b.addAssociate(new Associate(7));
                                //Line 5
  b.showAssociate();            //Line 6
  a=b;                          //Line 7
  b.removeAssociate();          //Line 8
  a.showAssociate();            //Line 9
}
```

Part A). Circle the line numbers, if any, that will cause the creation of an alias via pointers;

ANSWER:	1	2	3	4	5	6	7	8	9	none

Part B). Circle the line numbers, if any, that will cause the creation of a dangling pointer; i.e., a pointer to an address that is no longer owned/reserved by the program.

ANSWER:	1	2	3	4	5	6	7	8	9	none

Part C). Circle the line numbers, if any, that will cause a memory leak; i.e., the creation of memory owned/reserved by the program but inaccessible to it.

ANSWER:	1	2	3	4	5	6	7	8	9	none

Part D). Circle the line numbers, if any, that will logically cause an illegal access; i.e., an attempt to access an address not owned/reserved by the program.

ANSWER:	1	2	3	4	5	6	7	8	9	none

Part E). Which of the following is printed by line 2?

A). Hmm, value is 0
B). Hmm, value is 42
C). Hmm, value is 17
D). None of these

ANSWER:

Question No. 2: **(**14**)**

Consider the following program and answer questions from Part-A to Part-G:

```
class B;     //Line 1
class A
{
public:
	A();
	A(int);
	A(const A &);
	A& operator = (const A &);
	void operator = (const B &);
	~A();
	A(const B &);
	operator B();
};
class B
{
public:
	B();
	B(int);
	B(const B &);
	B& operator = (const B &);
	void operator = (const A &);
	~B();
	B(const A &);
	operator A();
};
A::A()
{
	cout<<"\nA::A()";
}
A::A(int)
{
	cout<<"\nA::A(int)";
}
A::A(const A &)
{
	cout<<"\nA::A(const A &)";
}
A& A::operator = (const A&)
{
	cout<<"\nA::operator =(const A &)";
	return *this;
}
void A::operator = (const B&)
{
	cout<<"\nA::operator =(const B &)";
}
A::~A()
{
	cout<<"\nA::~A()";
}
A::A(const B &)
{
	cout<<"\nA::A(const B &)";
}
A::operator B()
{
	cout<<"\nA::operator B()";
	return B();
}
B::B()
{
	cout<<"\nB::B()";
}
B::B(int)
{
	cout<<"\nB::B(int)";
}
B::B(const B &)
{
	cout<<"\nB::B(const B &)";
}
B& B::operator = (const B &)
{
	cout<<"\nB::operator = (const B &)";
	return *this;
}
void B::operator = (const A &)
{
	cout<<"\nB::operator =(const A &)";
}
B::~B()
{
	cout<<"\nB::~B()";
}
B::B(const A &)
{
	cout<<"\nB::B(const A &)";
}
B::operator A()
{
	cout<<"\nB::operator A()";
	return A();
}
void display( A ax, B bx)
{
	cout<<"\ndisplay(A,B)";
}
```

Part A). What is the relationship between class 'A' and class 'B'? **(**02**)**

Part B). What is the meaning of statement given in line-1? **(**02**)**

Part C). Give output of the following code? **(**03**)**

```
void main()
{
      A a1;
      B b1;
      display(a1,b1);
}
```

Part D). Give output of the following code? **(**03**)**

```
void main()
{
      A a1;
      a1 = 31;
}
```

Part E). Give output of the following code? **(**03**)**

```
void main()
{
```

```
        A a1;
        B b1;
        a1=b1;
}
```

Part F). Give output of the following code? **(**02**)**

```
void main()
{
        A a1;
        B b1;
        a1=(A)b1;
}
```

Part G). Consider now that the keyword 'explicit' is written before constructor '`A(const B &)`' then give output of the code written in Part-F? **(**02**)**

Part H). Consider now that there is no '`void operator = (const B &)`' declared and defined in class 'A' then give output of the code written in Part-F? **(**02**)**