

Soln.

FINAL EXAM

①

BS SE(A)

OOP THEORY

19K-1043

Q # 1

1.

(a). The error is the one additional bracket in 'cout' statement

(b)

```
template < typename T >
void print_max (const T& a, const T& b)
{   cout << (a > b) ? a : b << endl; }
```

2.

a) We can access the variable 'x' in function set by making set function a Friend function

```
class A {
    int x;
    public:
        display() { cout << x << endl; }
        friend void set();
};

void set() {
    A a; a.x = 10; a.display();
}
```

(2)

Alor

19K-1043

3.

a). Error is this that class B can not access the variable x of class A.

b). class A {

int x;

display() { cout << x; }

friend class B;

};

class B {

A a;

display() { cout << a.x; }

};

4.

a). class shape {

int x;

public:

Shape(int a): x(a) { }

};

(3)

Sol.

IAK-1043

```
class circle: virtual public shape {  
    int y;  
public:  
    Circle (int a, int b): y(a), Shape(b)  
    { }  
};
```

```
class Square: virtual public shape {  
    int z;  
public:  
    Square (int a, int b): z(a),  
    shape (b) { } };
```

```
class Circle-on-square: public circle,  
    public square  
{  
    int a;
```

```
public:  
    Circle-on-square (int x, int y, int z,  
    int a): a(a), circle (y, x),  
    Square (z, x) { }
```


④

Sl- 19K-1043

Q #2

(a)

Answer: One of the major criticism on friend is that it can have the access of all data whether private or protected which is against the method of encapsulation for hiding relevant data from others.

(b)

Answer: Because the object of abstract class can not be made and an abstract is a base class usually so it may have some features of which will be needed in child. So for that purpose partial virtual function is required and through this "In main, a pointer of abstract class can be made". These are the purpose of partial virtual function.

(5)

Solman

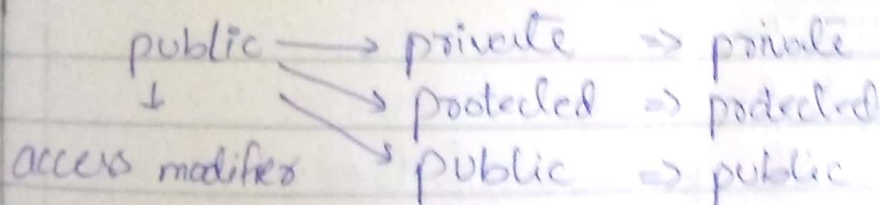
19K-1043

(c)

Answers: This function is not a pure virtual function as it requires to be initialized with zero. Therefore, this virtual function will not make class abstract.

(d)

Answers: If a child class inherits its parent class using public access modifier then the methods and members of parent's public data will become public in child also.



(e)

Answers: We cannot overload a function on return type because it is the last statement to be executed, furthermore it will completely change the meaning of function. That's why it is not done on return type.

(6)

Ans 19K-1043

(f)

Answer: The extension of C++ is not require because it replaces the line containing the #include header files with its complete content of the files included. In memory, the files are converted in object file which have every content of directives in the used file.

Q # 4

```
class BlueTech {  
    protected:  
        int year-of-manufactured;  
        int model;  
    public: int year (void) { return (this->year-of-manufactured); }  
        BlueTech (int y, int m) {  
            year-of-manufactured = y;  
            model = m; }  
};  
class led: public BlueTech {  
    private:  
        int screensize;  
        int supnumaps;
```

Soln

7
19K-1043

public:

```
led (int a, int b, int c, int d):  
    screensize (a), supramaps (b),  
    BlueTech (c, d) { }
```

};

```
class Mobile: virtual public BlueTech  
{
```

private:

```
    int camerareolution;
```

public:

```
    Mobile (int a, int b, int c):
```

```
        camerareolution (a), BlueTech (a, b) { }
```

```
    int year (void) {
```

```
        return (this->year-of-manufacture);
```

```
    }
```

};

```
class Tablet: public BlueTech {
```

```
    int screensize;
```

public:

```
    Tablet (int a, int b, int c):
```

```
        screensize (a), BlueTech (a, b) { }
```

};

Solu

(8)

191K-1043

```
class Tracking : virtual public BlueTech
{
    int accuracy;
public:
    Tracking (int a, int b, int c) :
        accuracy(a), BlueTech(b, c) {}
    int year (void) {
        return (this->year-of-manufacture);
    }
}
friend class RedTech;
```

```
class SmartRing : public Mobile, public
                    Tracking
{
public:
    SmartRing (int a, int b, int c, int d)
        : Mobile(a, b, c), Tracking(b, d) {}
};
```

```
class RedTech {
public:
    void checker (Tracking &T) {
        try {
            if (T.year-of-manufacture
                > 2010)
                cout << "ok";
        }
    }
};
```


(9)

Soln

19K-1043

th
else

```
throw (T. year-of-manufactured);  
{ catch (T. year-of-manufactured) {  
    cout << "earlier"; }  
};
```

```
template < class T >  
void item-sort (T &a, T &b, T &c) {
```

```
    if ( int x, y, z;  
        x = a.year(); y = b.year();  
        z = c.year();  
        if ( a > b x > y && x > z ) {  
            cout << "a is recent"; }  
        if ( y > x && y > z ) {  
            cout << "b is recent"; }  
        if ( z > x && z > y ) {  
            cout << "c is recent"; }  
        else
```

```
            cout << "Other are not  
            recent";
```

```
};
```

(10)

Salman

19K-1043

Q # 3

```
class Emp {  
    int ID; static int count;
```

```
public:
```

```
    Emp (int a): ID (a) { }
```

```
    virtual void post_vacancy () = 0;
```

```
    friend class candidate;
```

```
    void receive_app (candidate & c);
```

```
    void select (candidate & c);
```

```
    ~Emp () { count++; }
```

```
    void operator < (candidate & c);
```

```
};
```

```
class Edu : public Emp {  
    int ncampus;
```

```
public:
```

```
    Edu (int a, int b) : ncampus (a),
```

```
        Emp (b) { cout << "
```

```
teach
```

```
void post_vacancy (void) {
```

```
    cout << "teaching years and
```

```
    ability to cope with pressure"; }
```


(11)

Soln:

19/11-10/43

```
class Pharma: public Emp {  
    int annual bud;  
public:  
    Pharma (int a, int b) : annual bud(a),  
        Emp (b) { }  
    void post-vacancy (void) {  
        cout << "good analytical skill"; }  
};
```

```
class Bank: public Emp {  
    int branch;  
public:  
    Bank (int a, int b) : branch(a),  
        Emp (b) { }  
    void post-vacancy (void) {  
        cout << "Good communication skill"; }  
};
```

```
class Const: public Emp {  
    int project;  
public:  
    Const (int a, int b) : project(a),  
        Emp (b) { }  
    void post-vacancy (void) {  
        cout << "ability to work in  
            remote areas"; }  
};
```

Salun

19K-1043

class Candidate

void Emp::operator < (candidate &c)

{

if (this->^{exp}~~candidate~~ < c.~~exp~~)
 cout << "High experience";

else

cout << "low experience";

}

void Emp::receive_app (candidate &c) {

cout << "requirement filled";

select (candidate &c);

}

void Emp::select (candidate &c) {

this < c ; }

class Candidate {

string name, Qual, DOB, add ;

int Salary, id, exp ;

public:

 Candidate (string n, string q,
 string d, string ad, int s, int id, int ex)
 name(n), Qual(q), DOB(d), add(ad),
 Salary(s), Id(id), exp(ex) { }

(13)

Sulmu

19K-1043

```
Emp e(id);  
e.receive_app(this);
```

```
}
```

```
class Moderator {
```

```
public:
```

```
void write_data(void) {
```

```
    fstream fp;
```

```
    fp.open("info.txt", ios::app);
```

```
    fp << count;
```

```
    fp.close();}
```

```
void write_data(string m) {
```

```
    fstream fp;
```

```
    fp.open("message.txt", ios::app);
```

```
    fp << m;
```

```
    fp.close();
```

```
};
```

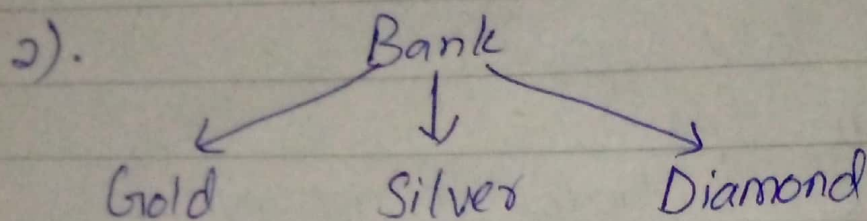

(14)

Salaries

19K-1043

Q # 5

1). Classes 4 Bank, Diamond, Silver, Gold



Multilevel inheritance

3). Gold member income transition is 1 lac. Limit must be 150,000 per day, balance greater or equal to 10,000.

Silver income at least 20,000 lac
Limit 100,000 per day
Balance 10,000 at least less than equal

Diamond income 200,000 Limit 250,000 per day. Balance 50,000.

Soln

19K-1043

```
Gold: try {  
    if (!withdrawal <= 150,000)  
        throw (withdrawal);  
    if (!transact >= 100,000)  
        throw (transact);  
    if (!balance >= 100,000)  
        throw (balance);  
} catch (Exception e) {  
    cout << e.what(); }  
}
```

```
Diamond: try {  
    if (!balance >= 50,000)  
        throw (balance);  
    if (!transact >= 250,000)  
        throw (transact);  
    if (!balance ! withdrawal >= 200,000)  
        throw (withdrawal);  
} catch (Exception ex) {  
    cout << ex.what(); }  
}
```

```
Silver : try { if (!balance < 100,000)  
                throw (balance);  
    if (!transaction < 200,000)  
        throw (transaction);  
    if (!withdrawal < 100,000)  
        throw (withdrawal);  
} catch (Exception ex)  
    cout << ex.what();
```

(16)

Soln 19 K-1043

// friend function for diamond

4). void POS (void){

if (balance \geq 10,000)

cout << "loan granted
Diamond";

}