## Question : 1

## A)

**i) Static Binding**

If we inherit a class into the derived class and provide a definition for one of the base class

function again inside the derived class, then that function is said to be overridden . This is usully called function overiding , there are basically two ways of doing that :

Connecting the function call to the function body is called Binding. When it is done before

the program is run, its called  **Static Binding**.

But when we use a Base class&#39;s pointer or reference to hold Derived class object, then

Function call Binding gives some unexpected results. Then there comes a use of late binding or dynamic binding

**Dynamic Binding**

In Late Binding function call is resolved at runtime. Hence, now compiler determines the

type of object at runtime, and then binds the function call. Late Binding is also

called Dynamic Binding TO attain this binding , there comes a use of the word **virtual** which is used to do the late binding.

**ii)**

Operator overloading is basically comes when we want to do perform some operations with the objects.Operator overloading is an important concept in C++. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc.

**iii)**

The main difference between concrete and an abstract clsss is that concrete class can have instatiation of object while this feature is not in there in abstract class . An abstract class cannot be instantiated because at least one method has not been implemented . They are meant to extended , like its implementation can be used by its child classes .

In abstract class there is must a pure virtual function should be declared and cannot be defined within the class scope and the child class of that abstract class must override that pure virtual function otherwise it would become also an abstract class, while in the concrete class there no such type of pure virtual function

**iv)**

To make a function read only we have to put keyword ‘const’ before it. As it will restrict it to only read not to change any function by other operations of compiler or user. In error checking code, const member functions and if/else condition can be defined.

### B)

**CODE:**

**#include <iostream>**

**using namespace std;**

**class Person{**

**string name;**

**int age;**

**string gender;**

**public:**

**Person() {};**

**Person(string n,int a,string g):name(n),age(a),gender(g)**

**{**

**};**

**string getName()**

**{**

**return name;**

**}**

**int getAge()**

**{**

**return age;**

**}**

**string getGender()**

**{**

**return gender;**

**}**

**void setName(string n)**

**{**

**name=n;**

**}**

**void setAge(int a)**

**{**

**age=a;**

**}**

**void setGender(string g)**

**{**

**gender=g;**

**}**

**void eat()**

**{**

**cout<<"I am a person and I can eat eating"<<endl;**

**}**

**void walk()**

**{**

**cout<<"I am a person and i can walk"<<endl;**

**}**

**};**

**class Student:public Person**

**{**

**string program;**

**int studyyear;**

**public:**

**Student() {};**

**Student(string p,int s,string n,int a,string g):Person(n,a,g)**

**{**

**program=p;**

**studyyear=s;**

**}**

**string getProgram()**

**{**

**return program;**

**}**

**int getStudyYear()**

**{**

**return studyyear;**

**}**

**void setProgram(string p)**

**{**

**program=p;**

**}**

**void setStudyYear(int s)**

**{**

**studyyear=s;**

**}**

**void study()**

**{**

**cout<<"I am A student currently and I am studying "<<endl;**

**}**

**void heldExam()**

**{**

**cout<<"Exam are just aound the corner"<<endl;**

**}**

**};**

**class Teacher:public Person**

**{**

**private:**

**string designation;**

**int salary;**

**public:**

**Teacher() {};**

**Teacher(string d,int sa,string n,int a,string g):Person(n,a,g)**

**{**

**designation=d;**

**salary=sa;**

**}**

**string getDesignation()**

**{**

**return designation;**

**}**

**int getSalary()**

**{**

**return salary;**

**}**

**void setDesignation(string d)**

**{**

**designation=d;**

**}**

**void setSalary(int sa)**

**{**

**salary=sa;**

**}**

**void teach()**

**{**

**cout<<"Hi I am a Teacher and I teach students"<<endl;**

**}**

**void takeExam()**

**{**

**cout<<"I am a strict checker in exams "<<endl;**

**}**

**};**

**class Doctor {**

**string designation;**

**double salary;**

**public:**

**Doctor(){ };**

**Doctor(string d, double s): designation(d), salary(s) {};**

**string getDesignation() {**

**return designation;**

**}**

**double getSalary(){**

**return salary;**

**}**

**void setDesignation(string d) {**

**designation = d;**

**}**

**void setSalary(double s) {**

**salary = s;**

**}**

**void Checkup() {**

**cout << "I reommend my patients to have a checkup in every mnonth!" << endl;**

**}**

**void Prescribe() {**

**cout << "I try to prescribe best medicines for my patients!" << endl;**

**}**

**};**

**int main()**

**{**

**cout<<"\tName :\tRehan Mumtaz \n\tRoll #:\tSE-19036";**

**Student ss("Software Engineering",3,"Rehan",20,"Male");**

**cout << "Student details:" << endl;**

**ss.eat();**

**ss.heldExam();**

**ss.study();**

**cout << endl;**

**Teacher tt("20 grade",120000,"MS.Maria",40,"Female");**

**cout << "Teacher details:" << endl;**

**tt.walk();**

**tt.teach();**

**cout << endl;**

**Doctor dd("19 grade", 140000);**

**cout << "Doctor details:" << endl;**

**dd.Checkup();**

**dd.Prescribe();**

**}**

**Output:**

