Lab Sheet 6
Understanding the Concept of Type Conversion and Inheritance

Type Conversion

To convert data from basic type to user defined type and vice versa we cannot rely on built in conversion routines because compiler doesn't know anything about user defined types. We have to provide the conversion routines to be used for type casting. Following methods are used for conversion.

a) To convert from basic type to user defined type, conversion is done by using the constructor function with one argument of basic type as follows

```
MyClass(BasicType var)
{

          //Conversion code

       }
```

b) To convert from user defined type to basic type, conversion is done by overloading the cast operator of basic type as a member function as follows

```
class MyClass
{

          ...

          Public:

               operator BasicType()

               {

                  //Conversion code

               }

      };
```

The conversions between objects of different classes can be carried out by using similar methods for conversions between basic types and user-defined types. For more details please refer to any text books.

Inheritance

Inheritance is the concept by which the properties of one entity are made available to another. It allows new classes to be built from older and less specialized classes instead of being rewritten from scratch. The class that inherits properties and functions is called the *subclass* or the *derived class* and the class from which they are inherited is called the *super class* or the *base class*. The derived class inherits all the properties of the base class and can add properties and refinements of its own. The base class remains unchanged.

Example

```
class person  //base class
{

protected:

   int age;

   char name[20];

public:

   void readAge(void)
   {
     cout<<"Enter Age: ";

     cin>>age;

   }

   void readName(void)
   {
     cout<<"\nEnter Name: ";

     cin>>name;

   }

   void printPerInformation(void)
   {
     cout<<"Name    - "<<name;

     cout<<"\nAge   - "<<age;

   }

};

//derived class inherits base class

class student:public person
{

private:

       int Sno;

       int percentage;

public:

       void readSno(void)
       {
         cout<<"Enter Sno.: "; cin>>Sno;
       }

    void readpercentage(void)
    {
      cout<<"Enter percentage: ";

      cin>>percentage;

    }

    void printStuInformation(void)
    {
      cout<<"\nName  - "<<name;

      cout<<"\nAge   - "<<age;

      cout<<"\nS.no  - "<<Sno<<endl;

      cout<<"Percentage- "<<percentage<<endl;

      cout<<"conclusion"<<endl;

      if(percentage>=80)

        cout<<"\nThe student is
Outstanding"<<endl

      else if(percentage>=70)

        cout<<"The student is Medium"<<endl;

      else

        cout<<"The student is Poor"<<endl;

    }

};

int main(void)
{
       clrscr();

       student st;

       st.readName();

       st.readAge();

       st.readSno();

       st.readpercentage();

       st.printStuInformation();

       return 0;

}
```

In Above example, person is the base class whereas student is the derived class which inherits all the features of the base class. Multiple classes may be derived from same base class and a derived class can also inherit characteristics of two or more classes.

This pointer

A special pointer called `this` pointer can be employed in C++ programs. When the object needs to point to itself then `this` pointer is used. Remember `this` pointer points (or represents the address of the) object of the class but not the class.

Example

```
class student:public person

{

private:

        ......

public:

        ......

        void printAddress(void)

        {

                cout<<"I am from within the object and my address is"<<this;

        }

};
```

Exercises

1. Write a program that can convert the Distance (meter, centimeter) to meters measurement in float and vice versa. Make a class distance with two data members, meter and centimeter. You can add function members as per your requirement.

2. Write two classes to store distances in meter-centimeter and feet-inch system respectively. Write conversions functions so that the program can convert objects of both types.

3. Create a class called Musicians to contain three methods string ( ), wind ( ) and perc ( ). Each of these methods should initialize a string array to contain the following instruments
   - veena, guitar, sitar, sarod and mandolin under string ( )
   - flute, clarinet saxophone, nadhaswaram and piccolo under wind ( )
   - tabla, mridangam, bangos, drums and tambour under perc ( )

It should also display the contents of the arrays that are initialized. Create a derived class called **TypeIns** to contain a method called get ( ) and show ( ). The get ( ) method must display a menu as follows

Type of instruments to be displayed
a. String instruments

b. Wind instruments
c. Percussion instruments

The show ( ) method should display the relevant detail according to our choice. The base class variables must be accessible only to its derived classes.

4. Write three derived classes inheriting functionality of base class **person** (should have a member function that asks to enter name and age) and with added unique features of **student, and employee**, and functionality to assign, change and delete records of student and employee. And make one member function for printing address of the objects of classes (base and derived) using this pointer. Create two objects of base class and derived classes each and print the addresses of individual objects. Using calculator, calculate the address space occupied by each object and verify this with address spaces printed by the program.

5. Write base class that ask the user to enter a complex number and make a derived class that adds the complex number of its own with the base. Finally make third class that is friend of derived and calculate the difference of base complex number and its own complex number.