Software Development Lab – II [15B17CI271]
Assignment Sheet
Week 4

COURSE OUTCOMES		COGNITIVE LEVELS
C173.1	Write programs in C++ to implement OOPs concepts related to objects, classes, constructor, destructor, and friend function.	Apply Level (Level 3)
C173.2	Write programs in C++ using OOPs concept like encapsulation, inheritance, polymorphism and abstraction.	Apply Level (Level 3)
C173.3	Write programs in C++ using Standard Template Library.	Apply Level (Level 3)
C173.4	Perform exception handling in C++ programs.	Apply Level (Level 3)
C173.5	Write MySQL queries to perform operations like ADD, DELETE, UPDATE, SELECT on relational databases.	Apply Level (Level 3)

Note: Students are advised to submit their solutions to respective lab faculty. The solution file must be named as "rollno_first name_w4.doc" (here w4 represents week4).

Q1. Write a program in C++ to override operator '+' to add two complex numbers.

Q2. For the following program, write a program in C++ to define the '<' operator for comparing the two objects of Box class.

```
#include <iostream>
#include <string>
using namespace std;
class Box
{
        int capacity;
    public:
        Box(){}
        Box(double capacity){
                this->capacity = capacity;
        }
};

int main(int argc, char const *argv[])
{
        Box b1(10);
        Box b2 = Box(14);
        if(b1 < b2){
                cout<<"Box 2 has large capacity.";
        }
        else{
                cout<<"Box 1 has large capacity.";
        }
        return 0;
}
```

Q3. Overload the post & pre-increment operators, i.e. ‘x++’ and ‘++x’ such that it returns the output.

Q4. For the following inheritance code in C++, predict the outcome.

```
#include <iostream>
using namespace std;
class Teacher {
public:
  Teacher(){
    cout<<"Hey Guys, I am a teacher"<<endl;
  }
  string collegeName = "Beginnersbook";
};
//This class inherits Teacher class
class MathTeacher: public Teacher {
public:
  MathTeacher(){
    cout<<"I am a Math Teacher"<<endl;
  }
  string mainSub = "Math";
  string name = "Negan";
};
int main() {
  MathTeacher obj;
  cout<<"Name: "<<obj.name<<endl;
  cout<<"College Name: "<<obj.collegeName<<endl;
  cout<<"Main Subject: "<<obj.mainSub<<endl;
  return 0;
}
```

Q5. Imagine a publishing company that markets both book and audio cassette versions of its works. Create a class publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data. Write a main() program to test the book and tape classes by creating instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata().

Q6. Start with the publication, book, and tape classes of Q5. Add a base class sales that holds an array of three floats so that it can record the dollar sales of a particular publication for the last three months. Include a getdata() function to get three sales amounts from the user, and a putdata() function to display the sales figures. Alter the book and tape classes so they are derived from both publication and sales. An object of class book or tape should input and output sales data along with its other data. Write a main() function to create a book object and a tape object and exercise their input/output capabilities.

Q7. Assume that a bank maintains two kinds of accounts for customers, one called as savings account and the other as current account. The savings account provides simple interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Q8. Create a class account that stores customer name, account number and type of account. From this derive the classes cur_acct and sav_acct to make them more specific to their requirements. Include necessary member functions in order to achieve the following tasks:

- Accept deposit from a customer and update the balance.
- Display the balance.
- Compute and deposit interest.
- Permit withdrawal and update the balance.
- Check for the minimum balance, impose penalty, necessary and update the balance.

Do not use any constructors. Use member functions to initialize the class members.

Q8. Execute the following program and check the output

```
#include<iostream>
using namespace std;

class base {
        int arr[10];
        int f;
};

class b1: private base {int c; };

class b2: private base {int d; };

class derived: public b1, public b2 {};

int main(void)
{
derived d;
cout << sizeof(d);
return 0;
}
```