C++ LAB MANUAL

PART-A

1. **Write C++ program to design a class called BankAccount. Include following data members like name of the depositor, account number and balance. Use following member functions a) to initialize values b) deposit an amount c) to withdraw an amount d) to display name and balance.**

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

class Bank
{
    char name[10];
    int balance;
    int accno;

    public:
        void read(){
            cout<<"Enter the Name of the customer:";
            cin>>name;
            cout<<"Enter the accno of the customer:";
            cin>>accno;
            cout<<"Enter the balance amount:";
            cin>>balance;
        }

        void deposit(){
            int amt;
            cout<<"Enter the Amount to be deposited:";
            cin>>amt;
            balance=balance+amt;
            cout<<"Available balance is: "<<balance<<endl;
        }
        void withdraw(){
            int amt;
            cout<<"Enter the amount to withdraw:";
            cin>>amt;
            if((balance-amt)<500)
            {
                cout<<"insufficient balance to withdraw:"<<endl;
            }
            else
            {
                balance=balance+amt;
                cout<<"Available balance is: "<<balance;
            }
        }

        void display(){
            cout<<"Customer name: "<<name<<endl;
            cout<<"Customer account number: "<<accno<<endl;
            cout<<"Available balance: "<<balance<<endl;
        }
};

int main()
{
    Bank B;
    cout<<"------Read Customer Details------"<<endl;
    B.read();
    cout<<"------Display Customer Details------"<<endl;
    B.display();
    cout<<"------Deposit------"<<endl;
    B.deposit();
    cout<<"------Withdraw------"<<endl;
    B.withdraw();
    return 0;
}
```

2. **Write a C++ program to create a class called COMPLEX and implement the following overloading function ADD that return a COMPLEX number.**

i. ADD(a,c2);- where a is an integer(real part) &amp; c2 is a complex no.
ii. ADD(c1,c2); where c1 &amp; c2 are complex nos. use Function overloading (ADD) and Friend function concept for the implementation

```
#include <iostream>
using namespace std;
class Complex
{
    int real;
    double imag;
    public:
        void read(int r,double i)
        {
            real=r;
            imag=i;
        }
        void disp(){
            cout<<"Complex Number is: "<<real<<"+"<<imag<<"i"<<endl;
        }
        friend Complex Add(int a, Complex c)
        {
            Complex temp;
            temp.real = a + c.real;
            temp.imag = c.imag;
            return temp;
        }
        friend Complex Add(Complex c,Complex c1)
        {
            Complex temp;
            temp.real=c.real+c1.real;
            temp.imag=c.imag+c1.imag;
            return temp;
        }
};

int main()
{
    Complex C1,C2,C3,C4;
    C1.read(1,2.5);
    C1.disp();
    C2.read(2,1.5);
    C2.disp();
    C3 = Add(2,C1);
    C3.disp();
    C4 = (C3,C2);
    C4.disp();
    return 0;
}
```

3. **Write a C++ program with class Time with data members that represents hours and minutes. Include appropriate member functions to compute time in hours and minutes. (Use of objects as arguments).**

```
#include <iostream>

using namespace std;

class Time
{
    int Hr,Hr1,min,min1;
    public:

        void read()
        {
            cout<<"Enter time1: ";
            cin>>Hr>>min;
            cout<<"Enter time2: ";
            cin>>Hr1>>min1;
        }
        void AddTime()
        {
            int h=Hr+Hr1;
            int m=min+min1;

            if(m>=60)
            {
                h=h+1;
            m=m-60;
        }
        if(h>12)
        {
            h=h-12;
        }
        cout<<"Added Time is: "<<h<<":"<<m;
    }
};
int main()
{
    Time T;
    T.read();
    T.AddTime();
    return 0;
}
```

4. **Given that an EMPLOYEE class contains following members. Data members: Eno, Ename and salary Member functions: to read the data, to print data members. Write a C++ program to read the data of N employees and display details of each employee. (use Array of objects concept).**

```
#include <iostream>

using namespace std;

class Employee
{
        int Eno;
        char Ename[10];
        int salary;

public:
        void read() {
                cout<<"Enter the Employee number:";
                cin>>Eno;
                cout<<"Enter the employee name:";
                cin>>Ename;
                cout<<"Enter the employee salary:";
                cin>>salary;
        }

        void print() {
                cout<<"Employee Number:"<<Eno<<endl;
                cout<<"Employee Name:"<<Ename<<endl;
                cout<<"Employee Salary:"<<salary<<endl;
        }
};
int main()
{
        int n;
        cout<<"Enter the number of employee:";
        cin>>n;

        Employee E[n];
        cout<<"Enter The Empolyee Details"<<endl;
        for(int i=0; i<n; i++)
        {
                cout<<"--------Enter Employee "<<i+1<<" Details--------"<<endl;
                E[i].read();
        }
        cout<<"Employee Details Are"<<endl;
        for(int i=0; i<n; i++)
        {
                cout<<"--------Employee "<<i+1<<" Details--------"<<endl;
                E[i].print();
        }
        return 0;
}
```

5. **Write a C++ program to demonstrate to uses of constructors in derived class concept. (Any inheritance you can use but constructors in base class should have at least one parameter.)**

```
#include <iostream>
using namespace std;
class Alpha {
   protected: int n1;
   public:
      Alpha(int x) {
        cout << "Alpha constructed" << endl; n1 = x;
      }
      void putAlpha() {
        cout << "n1: " << n1 << endl;

      }
};

class Beta {
    protected: int n2;
    public:
      Beta(int x) {
          cout << "Beta constructed" << endl;
          n2 = x;
      }
      void putBeta() {
          cout << "n2: " << n2 << endl;
      }
};

class Gama : public Alpha, public Beta {
    int n3;
    public:
       Gama(int x, int y, int z) : Alpha(x), Beta(y) {
           cout << "Gama constructed" << endl;
           n3 = z;
       }
       void putGama() {
           cout << "n3: " << n3 << endl;
       }
};

int main()
{
    Gama g1(10, 20, 30);
    g1.putAlpha();
    g1.putBeta();
    g1.putGama();
    return 0;
}
```

PART-B

6. Write a C++ program to create a class sample with integer, character and float data members. Demonstrate Constructor Overloading on this class with all types of constructors including default argument constructor.

```
#include <iostream>
using namespace std;
class Sample
{
        int i;
        char c;
        double d;
public:
        Sample()
        {
                i=0;
                c='\0';
                d=0.0;
        }
        Sample(int x,char y, double z)
        {
                i=x;
                c=y;
                d=z;
        }
        Sample(Sample &s)
        {
                i=s.i;
                c=s.c;
                d=s.d;
        }
        Sample(int x, double z, char y='s')
        {
                i=x;
                c=y;
                d=z;
        }
        void display()
        {
                cout<<"i= "<<i<<endl;
                cout<<"c="<<c<<endl;
                cout<<"f="<<d<<endl;
        }
};
int main()
{
        Sample s1;
        cout<<"S1: "<<endl;
        s1.display();
        Sample s2(10,'a',5.6);
        cout<<"S2: "<<endl;
        s2.display();
        Sample s3(30,4.54);
        cout<<"S3: "<<endl;
        s3.display();
        Sample s4(s2);
        cout<<"S4: "<<endl;
        s4.display();
        return 0;
}
```

7. **Write a C++ program to demonstrate the working of dynamic constructors using a class STRING with string as data member inside the class, include appropriate member functions to display the object data and a member function to concatenate two strings.**

```
#include <iostream>
#include<string.h>

using namespace std;

class String
{
        int length;
        char *name;
public:
        String() {}
        String(char s[])
        {
                length=strlen(s);
                name=new char[length+1];
                strcpy(name,s);
        }
        void join(String A,String B)
        {
                length=A.length+B.length;
                name=new char[length+1];
                name=strcpy(name,A.name);
                name=strcat(name,B.name);
```

```
	}
	void display()
	{
		cout<<"name= "<<name<<endl;
	}
};

int main()
{
	String s1("wel");
	String s2("fare");
	cout<<"s1: "<<endl;
	s1.display();
	cout<<"s2: "<<endl;
	s2.display();
	String s3;
	s3.join(s1,s2);
	cout<<"s3: "<<endl;
	s3.display();
	String s4("come");
	cout<<"s4: "<<endl;
	s4.display();
	String s5;
	s5.join(s1,s4);
	cout<<"s5: "<<endl;
	s5.display();
	String s6("done");
	cout<<"s6: "<<endl;
	s6.display();
	String s7;
	s7.join(s1,s6);
	cout<<"s7: "<<endl;
	s7.display();
	return 0;
}
```

8. Write a C++ program for the diagram using Hierarchical inheritance. Use your own data members and member functions to display student details.

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

class student
{
protected:
	char name[30];
	int age;
```

```
        int usn;
public:
        void getstudent()
        {
                cout<<"Enter name "<<endl;
                cin>>name;
                cout<<"Enter age "<<endl;
                cin>>age;
                cout<<"Enter usn "<<endl;
                cin>>usn;
        }
};

class medical:public student
{
        int year;
public:
        void getmedical()
        {
                cout<<"Enter year"<<endl;
                cin>>year;
        }
        void display()
        {
                cout<<"Medical student details: "<<endl;
                cout<<"Name: "<<name<<endl;
                cout<<"Age: "<<age<<endl;
                cout<<"USN: "<<usn<<endl;
                cout<<"Year: "<<year<<endl;
        }
};

class engineering:public student
{
        int sem;
        char branch [30];
public:
        void getengineering()
        {
                cout<<"Enter sem"<<endl;
                cin>>sem;
                cout<<"Enter branch"<<endl;
                cin>>branch;
        }
        void display()
        {
                cout<<"Engineering student details: "<<endl;
                cout<<"Name: "<<name<<endl;
                cout<<"Age: "<<age<<endl;
                cout<<"USN: "<<usn<<endl;
                cout<<"Semester: "<<sem<<endl;
                cout<<"Branch: "<<branch<<endl;
        }
};

int main()
{
        //medical
        medical m1;
        cout<<"Enter medical student details"<<endl;
        m1.getstudent();
        m1.getmedical();
        m1.display();

        //engineering
        engineering el;
        cout<<"Enter engineering student details"<<endl;
        el.getstudent();
        el.getengineering();
        el.display();

        return 0;
}
```

9. **Create a base class shape to store two double type values. Derive two specific classes called triangle and rectangle from the base shape. Add to the base class, a member function get_data() to initialize base class data members and another function display_area() to compute and display the area of figures. Make display_area() as a virtual function and redefine this function in the derived classes to suit the requirements. (Area of rectangle=x*y, Area of triangle=1/2*x*y).**

```
#include <iostream>

using namespace std;

class Shape {
   protected: double x, y;
   public:
      void get_data() {
         cout << "Enter the values for x and y: ";
         cin >> x >> y;
      }

      // Pure virtual function
      virtual void display_area() = 0;
};

class Triangle : public Shape {
    public:
        void display_area() override {
            cout << "The area of the triangle is: " << 0.5 * x * y << endl;
        }
};

class Rectangle : public Shape {
    public:
        void display_area() override {
            cout << "The area of the rectangle is: " << x * y << endl;
        }
};

int main()
{
    int ch;
    char c;

    do {
        cout << "Choose a shape to calculate area:\n";
        cout << "1. Triangle\n";
        cout << "2. Rectangle\n";
        cout << "Enter your choice (1 or 2): ";
        cin >> ch;
        Shape* shapePtr = nullptr;

        switch (ch) {
            case 1: shapePtr = new Triangle();
                    break;
            case 2: shapePtr = new Rectangle();
                    break;
            default: cout << "Invalid choice!" << endl;
                     continue;
        }

        shapePtr->get_data();
        shapePtr->display_area();
        delete shapePtr;
        cout << "Do you want to calculate another area? (y/n): ";
        cin >> c;

    } while (c == 'y' || c == 'Y');

    return 0;
}
```

10. Write a C++ program to overload binary + and operator to add and subtract two complex numbers. Define relevant data members and member functions for reading and displaying the complex objects.

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

class Complex
{
        int real;
        float imag;

public:
        void read(int r, float i)
        {
                real=r;
                imag=i;
        }

        void display()
        {
                cout<<real<<"+i"<<imag<<endl;
        }

        Complex operator+(Complex c)
        {
                Complex temp;
                temp.real-real+c.real;
                temp.imag-imag+c.imag;
                return temp;
        }

        Complex operator-(Complex c)
        {
                Complex temp;
                temp.real-real-c.real;
                temp.imag-imag-c.imag;
                return temp;
        }
};

int main()
{
        Complex c1,c2,c3,c4;
        c1.read(1,1.4);
        c2.read(3,2.2);
        cout<<"C1: "<<endl;
        c1.display();
        cout<<"C2: "<<endl;
        c2.display();
        c3=c1+c2;
        cout<<"C3:"<<endl;
        c3.display();
        c4=c2-c1;
        cout<<"C4: "<<endl;
        c4.display();
        return 0;
}
```