**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 BankAccount
 {
    char name[20];
    int accno;
   int balance;
public:
    void read()
    {
       cout<<"Enter name"<<endl;
       cin>>name;
       cout<<"Enter Accno"<<endl;
       cin>>accno;
       cout<<"Enter Balance"<<endl;
       cin>>balance;
    }
    void print()
    {
       cout<<"Name: "<<name<<endl;
       cout<<"Account Number: "<<accno<<endl;
       cout<<"Balance: "<<balance<<endl;
    }
    void deposit()
    {
       int amount;
       cout<<"Enter amount"<<endl;
       cin>>amount;
       balance=balance+amount;
    }
    void withdraw()
    {
       int amount;
       cout<<"Enter amount to withdraw"<<endl;
       cin>>amount;
       if(amount>balance)
       {
          cout<<"Insufficient"<<endl;
       }
       else
       {
          balance=balance-amount;
       }
    }
    void put_balance()
    {
       cout<<"Account balance is: "<<balance<<endl;
    }
```

```
};
int main()
{
  BankAccount b1;
  b1.read();
  b1.print();
  b1.deposit();
  b1.withdraw();
  b1.put_balance();
  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) & c2 is a complex no.**

**ii. ADD(c1,c2);- where c1 & c2 are complex nos.**

**use Function overloading(ADD) and Friend function concept for the implementation**

```
#include   <iostream>

using  namespace  std;

class COMPLEX
 {
private:int r, i;
public: void read();
                void disp();
                friend COMPLEX ADD(int x, COMPLEX c);
                friend COMPLEX ADD(COMPLEX c1,COMPLEX c2);
 };
 void COMPLEX::read()
 {
        cin>>r>>i;
 }
 void COMPLEX::disp()
 {
        cout<<r<<"+"<<i<<"i";
 }
 COMPLEX ADD(int x, COMPLEX c)
 {
        COMPLEX t;
        t.r=x+c.r;
        t.i=c.i;
        return t;
 }
```

```
COMPLEX ADD(COMPLEX c1,COMPLEX c2)
{
        COMPLEX t;
        t.r=c1.r+c2.r;
        t.i=c1.i+c2.i;
        return t;
}
int main()
{
        COMPLEX c1,c2,c3,c4;
        int a;
        cout<<"Enter 1st Complex no :";
        c1.read();
        cout<<"Enter 2nd Complex no :";
        c2.read();
        cout<<"Enter the value of a:";
        cin>>a;
        c3=ADD(a,c2);
        c4=ADD(c1,c2);
        cout<<"\n1st Complex no :";
        c1.disp();
        cout<<"\n2nd Complex no :";
        c2.disp();
        cout<<"\n\nADD(a,c2) :";
        c3.disp();
        cout<<"\nADD(c1,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 arugemnts).**

```
#include <iostream>

using namespace std;
class Time
 {
    int hours;
    int minutes;
 public:
    void read(int h,int m)
    {
       hours=h;
       minutes=m;
    }
    void put()
    {
```

```
        cout<<"hours: "<<hours<<endl;
        cout<<"minutes: "<<minutes<<endl;
    }
    void compute(Time t1,Time t2)
    {
        int h=(t1.minutes+t2.minutes)/60;
        minutes=(t1.minutes+t2.minutes)%60;
        hours=t1.hours+t2.hours+h;
    }
};
int main()
{
 Time A,B,C;
 A.read(4,40);
 B.read(4,40);
 cout<<"A: "<<endl;
 A.put();
 cout<<"B: "<<endl;
 B.put();
 C.compute(A,B);
 cout<<"C: "<<endl;
 C.put();
   return 0;
}
```

**4. Given that an EMPLOYEE class ontains 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 detials of each employee.(use Array of objects concept).**

```
#include <iostream>

using namespace std;
class Employee
 {

    char ename[20];
    int eno;
   int esalary;
public:
    void read()
    {
        cout<<"Enter name"<<endl;
        cin>>ename;
        cout<<"Enter enumber"<<endl;
        cin>>eno;
        cout<<"Enter esalary"<<endl;
        cin>>esalary;
```

```
    }
    void print()
    {
       cout<<"Name: "<<ename<<endl;
       cout<<"Enumber: "<<eno<<endl;
       cout<<"Esalary: "<<esalary<<endl;
    }

};
int main()
{
  Employee e[10];
  int n;
  cout<<"Enter number of employees"<<endl;
  cin>>n;
  for(int i=0;i<n;i++)
  {
     cout<<"Eneter "<<i+1<<"Employee details"<<endl;
     e[i].read();

  }
  for(int i=0;i<n;i++)
  {
     cout<<i+1<<"Employee details"<<endl;
     e[i].print();

  }
   return 0;
}
```

**5. Write a C++ program with two classes A and B with one integer data member in each class. Write member functions to read and display, place a friend function in these classes which takes the data members of these classes and computes maximum of two data members. Demonstrate using main function.**

```
#include <iostream>

using namespace std;
class B;
 class A
 {
   int n;
public:
    void read(int a)
    {
       n=a;
    }
    void display()
    {
```

```
        cout<<"value= "<<n<<endl;
    }
    friend void maximum(A obj1,B obj2);
};
class B
{
  int n;
public:
    void read(int a)
    {
        n=a;
    }
    void display()
    {
        cout<<"value= "<<n<<endl;
    }
    friend void maximum(A obj1,B obj2);
};
void maximum(A obj1,B obj2)
{
    if(obj1.n>obj2.n)
    {
        cout<<"maximum value is: "<<obj1.n<<endl;
    }
    else
    {
        cout<<"maximum value is: "<<obj2.n<<endl;
    }
}
int main()
{
  A a1;
  a1.read(20);
  B b1;
  b1.read(10);
  cout<<"A: "<<endl;
  a1.display();
  cout<<"B: "<<endl;
  b1.display();
  maximum(a1,b1);
  return 0;
}
```

**6. 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;
}
```

**7. Write a C++ program to create a class sample with integer ,character and float datamembers. 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;
}
```

**8. Write a C++ program to demonstrate the working of dynamic constructors using a class STRNG 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;
}
```