***** Assignment No. 1 *****

```
#include<iostream>
using namespace std;
class complex
{
   float i;
   float r;
public:
   complex()
   {
      r=0;
      i=0;
   }
   complex operator+(complex);
   complex operator*(complex);
   friend istream &operator>>(istream &input,complex &t)
   {
      cout<<"\n Enter the real part:";
      input>>t.r;
      cout<<"\n Enter the imaginary part:";
      input>>t.i;
   }
   friend ostream &operator<<(ostream &output,complex &t)
   {
      output<<t.r<<"+"<<t.i<<"i";
   }
};
    complex complex::operator+(complex c)
   {
      complex temp;
```

```
        temp.r=r+c.r;
        temp.i=i+c.i;
        return temp;
    }

    complex complex::operator*(complex c)
    {
        complex temp2;
        temp2.r=(r*c.r)-(i*c.i);
        temp2.i=(i*c.r)+(r*c.i);
        return temp2;
    }
int main()
{
        complex c1,c2,c3,c4;
        cout<<"\n Default constructor value:";
        cout<<c1;
        cout<<"\n enter the first number:";
        cin>>c1;
        cout<<"\n enter the second number:";
        cin>>c2;
        c3=c1+c2;
        c4=c1*c2;
        cout<<"\n The first number is:"<<c1;
        cout<<"\n The second number is:"<<c2;
        cout<<"\n The addition is:"<<c3;
        cout<<"\n The multiplication is:"<<c4;
        return 0;
}
```

***** Assignment No.2 *****

```
#include<iostream>                    //header file used for cin and cout
#include<string.h>                   //header file for string class
using namespace std;                 //refer cin and cout
#define max 100;
class per_info                       //stud_info is friend class of per_info
{
   string lic, dob, bldgrp;          //personal info variables
   public:
   per_info();                       //DECLARE DEFAULT CONSTRUCTOR//
   per_info(per_info &);             //DECLARE COPY CONSTRUCTOR//
   ~per_info()                       //DEFINITION AND DECLARATION OF DESTRUCTOR
   {
       cout<<"\nDESTRUCTOR IS CALLED!!!!!"<<endl<<"RECORD DELETED SUCCESSFULLY"<<endl;
   }
   friend class student;             //FRIEND FUNCTION DECLARATION
};
class student                        //DEFINITION OF STUDENT CLASS
{
   string name, address, year;                    //OBJECTS OF STRING CLASS
   char div;
   int roll_no;
   long mob;
   static int cnt;                                //  STATIC VARIABLE DECLARATION
   public:
   void create(per_info &);                       //TO CREATE DATABASE AND PASSED
REFERENCE OF PERSONAL INFO OBJECT
   void display(per_info &);                  //TO DISPLAY DATABASE

   inline static void inccnt()                //STATIC FUNCTION
```

```
  {
    cnt++;
  }
  inline static void showcnt()          //STATIC FUNCTION
                                        //INLINE FUNCTION
  {
    cout<<"\nTOTAL NO OF RECORDS ARE : "<<cnt;
  }
  student();                    //DEFAULT CONSTRUCTOR//
  student(student &);               //COPY CONSTRUCTOR OF STUDENT CLASS//
  ~student()                    //DESTRUCTOR OF STUDENT CLASS//
  {
    cout<<"\nDESTRUCTOR IS CALLED!!!"<<endl<<"RECORD DELETED SUCCESSFULLY"<<endl;
  }
};

int student::cnt;           //DEFINITION OF STATIC MEMBER//
student::student()          //CONSTRUCTOR DEFINITION//
{
      name="ANAGHA NIRGUDE";
      address="SR NO.81 BABBAR SOLANKI \nDIGHI, PUNE";
      year="SE COMPUTER";
      div='A';
      roll_no=21042;
      mob=942329999;
}
per_info::per_info()        //CONSTRUCTOR DEFINITION//
{
     lic="ABD45656";
     dob="02/11/1997";
```

```
      bldgrp="A-";
}
student::student(student &obj)              //DEFINITION OF COPY CONTRUCTOR OF STUDENT CLASS
{
    this->name=obj.name;                              //this is a pointer points to the object which invokes
it
    this->address=obj.address;    //this-> can be written as name
    this->year=obj.year;
    this->div=obj.div;
    this->roll_no=obj.roll_no;
    this->mob=obj.mob;
}
per_info::per_info(per_info &obj)          //DEFINITION OF COPY CONTRUCTOR OF PERSONAL CLASS
{
   lic=obj.lic;
   dob=obj.dob;
   bldgrp=obj.bldgrp;
}
//TO CREATE THE DATABASE
//DEFINTION OF CREATE FUNTION
void student::create(per_info &obj)
{
cout<<"\nNAME : "<<endl;
cin>>name;
cout<<"\nADDRESS : "<<endl;
cin>>address;
cout<<"\nDATE OF BIRTH : "<<endl;
cin>>obj.dob;
cout<<"\nYEAR : "<<endl;
cin>>year;
cout<<"\nDIVISION: "<<endl;
```

```
cin>>div;
cout<<"\nROLL NUMBER : "<<endl;
cin>>roll_no;
cout<<"\nBLOOD GROUP : "<<endl;
cin>>obj.bldgrp;
cout<<"\nLICEINCE NUMBER : "<<endl;
cin>>obj.lic;
cout<<"\nPHONE NUMBER : "<<endl;
cin>>mob;
}

//DEFINTION OF DISPLAY FUNCTION
//TO DISPLAY DATABASE
void student::display(per_info &obj)
{
cout<<"\n**********************"<<endl;
cout<<"\nNAME OF STUDENT : "<<name<<endl;
cout<<"\nADDRESS OF STUDENT : "<<address<<endl;
cout<<"\nDATE OF BIRTH : "<<obj.dob<<endl;
cout<<"\nYEAR : "<<year<<endl;
cout<<"\nDIVISION : "<<div<<endl;
cout<<"\nROLL NO : "<<roll_no<<endl;
cout<<"\nBLOOD GROUP : "<<obj.bldgrp<<endl;
cout<<"\nLICEINCE NUMBER : "<<obj.lic<<endl;
cout<<"\nPHONE NUMBER : "<<mob<<endl;
cout<<"\n**********************"<<endl;
}
int main()
{
 int n;                                  //COUNT OF NUMBER OF STUDENTS
 int ch;
```

```
char ans;


cout<<"\nENTER NO OF STUDENTS :"<<endl;
cin>>n;
cout<<"\n**********************"<<endl;
student *sobj=new student[n];
per_info *pobj=new per_info[n];
do
{
        cout<<"\n Menu \n 1. Create Database \n 2. Display Databse \n 3. Copy Constructor\n 4. Default Constructor \n 5. Delete (Destructor)";
        cout<<"\n Enter your Choice: ";
        cin>>ch;
        switch(ch)
        {
        case 1: // create database
        {
                for(int i=0;i<n;i++)
                {
                    sobj[i].create(pobj[i]);
                    sobj[i].inccnt();


                }
        }
                break;
        case 2:
                {
                        sobj[0].showcnt();
                         for(int i=0;i<n;i++)
                         {
                                 sobj[i].display(pobj[i]);
```

```
                        }
            }
            break;
             case 3: // Copy Constructor
                    {
                            student obj1;
                            per_info obj2;
                            obj1.create(obj2);
                            student obj3(obj1);
                            per_info obj4(obj2);
                            cout<<"\n Copy Constructor is called ";
                            obj3.display(obj4);
                    }
        break;
            case 4: // Default Constructor
                    {
                            student obj1;    //obj1 is invoking default constructor of class student
                            per_info obj2;  //obj2 is invoking default constructor of class personal
                            cout<<"\n Default Constructor is called ";
                            obj1.display(obj2);
                    }
                    break;
            case 5: // destructor is called
                       delete [] sobj;
                       delete [] pobj;
            }
            cout<<"\n Want to continue:(y/n)";
            cin>>ans;
     }while(ans=='y');
    return 0;}
```

***** Assignment No.3 *****

```
#include<iostream>
#include<string>
using namespace std;
class publication
{
private:
   string title;
   float price;
public:
   publication() //constructor created
   {
      title="";
      price=0.0;
   }
   void get_data()
   {
      cout<<"\n Enter title   ";
      cin>>title;
      cout<<"\n Enter price   ";
      cin>>price;
   }
   void put_data()
   {
      cout<<"\n *********";
      cout<<"\n Information";
      cout<<"\n The title is :  "<<title;
      cout<<"\n The price is :  "<<price;
   }
};
```

```
class book:public publication
{
private:
   int pages;
public:
   book() //default constructor created
   {
      pages=0;
   }
   void get_data()
   {
      publication::get_data();
      cout<<endl;
      cout<<"\n Enter pages for count  ";
      cin>>pages;
   }
   void put_data()
   {
      publication::put_data();
      try
      {
         if(pages<0)
            throw pages;
      }
      catch(int f)
      {
         cout<<"\n error : page not valid  "<<f;
         pages=0;
      }
      cout<<"\n pages are:  "<<pages;
   }
```

```
};
class tape: public publication
{
private:
   float playtime;
public:
   tape()           //default constructor created
   {
      playtime=0.0;
   }
   void get_data()
   {
      publication::get_data();
      cout<<"\n Enter play time of cassete  ";
      cin>>playtime;
   }
   void put_data()
   {
      publication::put_data();
      try
      {
         if(playtime<0.0)
            throw playtime;
      }
      catch(float r)
      {
         cout<<"\n Error: invalid playtime"<<playtime;
         playtime=0.0;
      }
      cout<<"\n playtime is:"<<playtime;
   }
```

```
};
int main()
{
   book b[10];    //array type object created
   tape t[10];
   int choice=0,bookcount=0,tapecount=0;
   cout<<"\n **********";
   do
   {
      cout<<"\n *****Publishing company*****";
      cout<<"\n 1.Add book";
      cout<<"\n 2.Add tape";
      cout<<"\n 3.Display book";
      cout<<"\n 4.Display tape";
      cout<<"\n 5.Exit...thank you";
      cout<<"\n Enter choice";
      cin>>choice;
      switch(choice)
      {
         case 1:
            {
              cout<<"\n *****";
               cout<<"\n Add book: ";
               b[bookcount].get_data();
               bookcount++;
               break;
            }
      case 2:
         {
            cout<<"\n *****";
            cout<<"\n Add tape: ";
```

```
            t[tapecount].get_data();
            tapecount++;
            break;
        }
    case 3:
        {
            cout<<"\n (books)";
            for(int j=0; j<bookcount; j++)
            {
                b[j].put_data();
            }
            break;
        }
    case 4:
        {
            cout<<"\n (tape) ";
            for(int j=0; j<tapecount; j++)
            {
                t[j].put_data();
            }
            break;
        }
    case 5:
        {
            cout<<"\n *****Program successfully ended  ";
        }
    default:
        {
            cout<<"\n Invalid input";
        }
}
```

```
}
   while(choice != 5);
 return 0;
}
```

***** Assignment No.4 *****

```
#include<iostream>
#include<fstream>
using namespace std;
class student
{
	int roll;
	char name[30];
	float percentage;
	public:
		void set_data()
		{
			cout<<"\n Enter the roll no. :";
			cin>>roll;
			cout<<"\n Enter the name: ";
			cin>>name;
			cout<<"\n Enter the percentage :";
			cin>>percentage;
		}
		void show_data()
		{
			cout<<"\n Entered name is: "<<name;
			cout<<"\n Entered roll is: "<<roll;
			cout<<"\n Entered percentage is :"<<percentage;
		}
};//class
int main()
{
	int n,i;
	student s;
```

```
        fstream f;
        f.open("abc.txt",ios::out);
        cout<<"\n How many students data do you want to enter";
        cin>>n;
        for(i=0;i<n;i++)
        {
                s.set_data();
                f.write((char*)&s,sizeof s);
        }
        f.close();
        f.open("abc.txt",ios::in);
        for(i=0;i<n;i++)
        {
                f.read((char *)&s,sizeof s);
                s.show_data();
        }
        return 0;

}
```

***** Assignment No.5 *****

```
#include<iostream>
using namespace std;
int n;
#define size 10
template<class T>
void sel(T A[size])
{
   int i,j,min;
   T temp;
   for(i=0;i<n;i++)
   {
      min=i;
      for(j=i+1;j<n;j++)
      {
         if(A[j]<A[min])
         min=j;
      }
      temp=A[i];
      A[i]=A[min];
      A[min]=temp;
   }


   cout<<"\nSorted array:";
   for(i=0;i<n;i++)
   {
      cout<<" "<<A[i];
   }
}
```

```
int main()
{
   int A[size];
   float B[size];
   int i;

   cout<<"\nEnter total no of int elements:";
   cin>>n;
   cout<<"\nEnter int elements:";
   for(i=0;i<n;i++)
   {
      cout<<"a["<<i<<"]=";
        cin>>A[i];
   }
   sel(A);

   cout<<"\nEnter total no of float elements:";
   cin>>n;
   cout<<"\nEnter float elements:";
   for(i=0;i<n;i++)
   {
      cout<<"a["<<i<<"]=";
       cin>>B[i];
   }
   sel(B);
}
```

**********Assignment No.6**********

```
#include<iostream>  //standard input output stream header file

#include<algorithm>  //The STL algorithm are generic because they can operate on a variety of data structures

#include<vector>     //The header file for the STL vector library is vector

using namespace std;

class Item

{

   public:

       char name[10];

       int quantity;

       int cost;

       int code;


       bool operator==(const Item& i1)   //Boolean operator allow you create more complex conditional statements

       {

         if(code==i1.code)   //operator will return 1 if the comparision i true, or 0 if the comparision is false

         return 1;

         return 0;

       }


       bool operator<(const Item& i1)

       {

         if(code<i1.code)   //operator will return 1 if the comparision i true, or 0 if the comparision is false

         return 1;
```

```
        return 0;
      }
};

vector<Item> o1;
void print(Item &i1);
void display();
void insert();
void search();
void dlt();

bool compare(const Item &i1, const Item &i2)
{
   if  (i1.name != i2.name) return i1.cost < i2.cost;
   return i1.cost < i2.cost;
   }

int main()
{
   int ch;
   do
   {

   cout<<"\n******Menu******";
   cout<<"\n1.Insert";
   cout<<"\n2.Display";
   cout<<"\n3.Search";
```

```
cout<<"\n4.Sort";
cout<<"\n5.Delete";
cout<<"\n6.Exit";
cout<<"\nEnter your choice: ";
cin>>ch;

switch(ch)
{
   case 1:
        insert();
        break;
   case 2:
        display();
        break;
   case 3:
        search();
        break;
   case 4:
        sort(o1.begin(),o1.end(),compare);
        cout<<"\n\n Sorted on cost";
        display();
        break;
   case 5:
        dlt();
        break;
   case 6:
        exit(0);
```

```
    }
}while(ch!=7);
  return 0;
}

void insert()
{
    Item i1;
    cout<<"\nEnter Item Name: ";
    cin>>i1.name;
    cout<<"\nEnter Item Quantity: ";
    cin>>i1.quantity;
    cout<<"\nEnter Item Cost: ";
    cin>>i1.cost;
    cout<<"\nEnter Item Code: ";
    cin>>i1.code;
    o1.push_back(i1);
}
void display()
{
    for_each(o1.begin(),o1.end(),print);
}

void print(Item &i1)
{
    cout<<"\n";
```

```
    cout<<"\nItem Name: "<<i1.name;
    cout<<"\nItem Quantity: "<<i1.quantity;
    cout<<"\nItem Cost: "<<i1.cost;
    cout<<"\nItem Code: "<<i1.code;
}

void search()
{
    vector<Item>::iterator p;
    Item i1;
    cout<<"\nEnter Item Code to search:";
    cin>>i1.code;
    p=find(o1.begin(),o1.end(),i1);
    if(p==o1.end())
    {
        cout<<"\nNot found.";
    }
    else
    {
        cout<<"\nFound.";
    }
}

void dlt()
{
    vector<Item>::iterator p;
    Item i1;
```

```
    cout<<"\nEnter Item Code to delete: ";
    cin>>i1.code;
    p=find(o1.begin(),o1.end(),i1);
    if(p==o1.end())
    {
       cout<<"\nNot found.";
    }
    else
    {
       o1.erase(p);
       cout<<"\nDeleted.";
    }
}
```

```
 #include<iostream>
#include<map>
#include<string>
using namespace std;
int main()
{
   typedef map<string,int> mapType;
   mapType populationMap;
   populationMap["Maharashtra"] = 7026357;
   populationMap.insert(pair<string, int>("Rajasthan", 6578936));
   populationMap.insert(pair<string, int>("Karnataka", 6678996));
   populationMap.insert(pair<string, int>("Punjab", 5789032));
   populationMap.insert(pair<string, int>("West Bengal", 6676291));

   mapType::iterator iter;
   cout<<"==========Population of states in India==========\n";
   cout<<"\n Size of populationMap"<<populationMap.size()<<"\n";
   string state_name;
   cout<<"\n Enter name of the state :";
   cin>>state_name;
   iter = populationMap.find(state_name);
   if(iter!= populationMap.end())
   cout<<state_name<<"'s population is "<<iter->second;
   else
   cout<<"Key is not populationMap"<<"\n";
   populationMap.clear();
}
```