**Program 1**

/\*

Implement a class Complex which represents the Complex Number data type. Implement the following

1. Constructor (including a default constructor which creates the complex number 0+0i).

2. Overload operator+ to add two complex numbers.

3. Overload operator\* to multiply two complex numbers.

4. Overload operators << and >> to print and read Complex Numbers

\*/

# include<iostream>

using namespace std;

class Complex //decaring Class Complex

{

double real;

double img;

public:

Complex(); // Default Constructor

friend istream & operator >> (istream &, Complex &); // Input

friend ostream & operator << (ostream &, const Complex &); // Output

Complex operator + (Complex); // Addition

Complex operator \* (Complex); // Multiplication

};

Complex::Complex() // Default Constructor

{

real = 0;

img = 0;

}

istream & operator >> (istream &, Complex & i)

{

cin >> i.real >> i.img;

return cin;

}

ostream & operator << (ostream &, const Complex & d)

{

cout << d.real << " + " << d.img << "i" << endl;

return cout;

}

Complex Complex::operator + (Complex c1) // Overloading + operator

{

Complex temp;

temp.real = real + c1.real;

temp.img = img + c1.img;

return temp;

}

Complex Complex::operator \* (Complex c2) // Overloading \* Operator

{

Complex tmp;

tmp.real = real \* c2.real - img \* c2.img;

tmp.img = real \* c2.img + img \* c2.real;

return tmp;

}

int main()

{

Complex C1, C2, C3, C4;

int flag = 1;

char b;

while (flag == 1)

{

cout << "Enter Real and Imaginary part of the Complex Number 1 : \n";

cin >> C1;

cout << "Enter Real and Imaginary part of the Complex Number 2 : \n";

cin >> C2;

int f = 1;

while (f == 1)

{

cout << "Complex Number 1 : " << C1 << endl;

cout << "Complex Number 2 : " << C2 << endl;

cout << "\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "1. Addition of Complex Numbers" << endl;

cout << "2. Multiplication of Complex Numbers" << endl;

cout << "3. Exit\n";

int a;

cout << "Enter your choice from above MENU (1 to 3) : ";

cin >> a;

if (a == 1)

{

C3 = C1+C2;

cout << "Addition : " << C3 << endl;

cout << "Do you wan to perform another operation (y/n) : \n";

cin >> b;

if (b == 'y' | | b == 'Y')

{

f=1;

}

else

{

cout << "Thanks for using this program!!\n";

flag=0;

f=0;

}

}

else if (a == 2)

{

C4 = C1 \* C2;

cout << "Multiplication : " << C4 << endl;

cout << "Do you wan to perform another operation (y/n) : \n";

cin >> b;

if (b == 'y' | | b == 'Y')

{

f=1;

}

else

{

cout << "Thanks for using this program!!\n";

flag=0;

f=0;

}

}

else

{

cout << "Thanks for using this program!!\n";

flag=0;

f=0;

}

}

}

return 0;

}

**Program 2**

**/\***

**Experiment Number 2 : Develop a program in C++ to create a database of**

**student’s information system**

**containing the following information: Name, Roll number, Class, Division,**

**Date of Birth, Blood group,**

**Contactaddress, Telephone number, Driving license no. and other. Construct**

**the database with**

**suitable member functions. Make use of constructor, default constructor,**

**copy constructor,**

**destructor, static member functions, friend class, this pointer, inline**

**code and dynamic**

**memory allocation operators-new and delete as well as exception handling.**

**\*/**

**#include<iostream>**

**#include<string.h>**

**using namespace std;**

**class StudData;**

**class Student{**

**string name;**

**int roll\_no;**

**string cls;**

**char\* division;**

**string dob;**

**char\* bloodgroup;**

**static int count;**

**public:**

**Student() // Default Constructor**

**{**

**name="";**

**roll\_no=0;**

**cls="";**

**division=new char;**

**dob="dd/mm/yyyy";**

**bloodgroup=new char[4];**

**}**

**~Student()**

**{**

**delete division;**

**delete[] bloodgroup;**

**}**

**static int getCount()**

**{**

**return count;**

**}**

**void getData(StudData\*);**

**void dispData(StudData\*);**

**};**

**class StudData{**

**string caddress;**

**long int\* telno;**

**long int\* dlno;**

**friend class Student;**

**public:**

**StudData()**

**{**

**caddress="";**

**telno=new long;**

**dlno=new long;**

**}**

**~StudData()**

**{**

**delete telno;**

**delete dlno;**

**}**

**void getStudData()**

**{**

**cout<<"Enter Contact Address : ";**

**cin.get();**

**getline(cin,caddress);**

**cout<<"Enter Telephone Number : ";**

**cin>>\*telno;**

**cout<<"Enter Driving License Number : ";**

**cin>>\*dlno;**

**}**

**void dispStudData()**

**{**

**cout<<"Contact Address : "<<caddress<<endl;**

**cout<<"Telephone Number : "<<\*telno<<endl;**

**cout<<"Driving License Number : "<<\*dlno<<endl;**

**}**

**};**

**inline void Student::getData(StudData\* st)**

**{**

**cout<<"Enter Student Name : ";**

**getline(cin,name);**

**cout<<"Enter Roll Number : ";**

**cin>>roll\_no;**

**cout<<"Enter Class : ";**

**cin.get();**

**getline(cin,cls);**

**cout<<"Enter Division : ";**

**cin>>division;**

**cout<<"Enter Date of Birth : ";**

**cin.get();**

**getline(cin,dob);**

**cout<<"Enter Blood Group : ";**

**cin>>bloodgroup;**

**st->getStudData();**

**count++;**

**}**

**inline void Student::dispData(StudData\* st1)**

**{**

**cout<<"Student Name : "<<name<<endl;**

**cout<<"Roll Number : "<<roll\_no<<endl;**

**cout<<"Class : "<<cls<<endl;**

**cout<<"Division : "<<division<<endl;**

**cout<<"Date of Birth : "<<dob<<endl;**

**cout<<"Blood Group : "<<bloodgroup<<endl;**

**st1->dispStudData();**

**}**

**int Student::count;**

**int main()**

**{**

**Student\* stud1[100];**

**StudData\* stud2[100];**

**int n=0;**

**char ch;**

**do**

**{**

**stud1[n]=new Student;**

**stud2[n]=new StudData;**

**stud1[n]->getData(stud2[n]);**

**n++;**

**cout<<"Do you want to add another student (y/n) : ";**

**cin>>ch;**

**cin.get();**

**} while (ch=='y' || ch=='Y');**

**for(int i=0;i<n;i++)**

**{**

**cout<<"---------------------------------------------------------------"<<endl;**

**stud1[i]->dispData(stud2[i]);**

**}**

**cout<<"---------------------------------------------------------------"<<endl;**

**cout<<"Total Students : "<<Student::getCount();**

**cout<<endl<<"---------------------------------------------------------------"<<endl;**

**for(int i=0;i<n;i++)**

**{**

**delete stud1[i];**

**delete stud2[i];**

**}**

**return 0;**

**}**

**Program 3**

**/\***

**Imagine a publishing company which does marketing for book**

**and audiocassette versions.**

**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).**

**Write a program that instantiates the book and tape classes,**

**allows user to enter data and**

**displays the data members.If an exception is caught, replace**

**all the data member values**

**with zero values.**

**\*/**

**# include<iostream>**

**# include<stdio.h>**

**using namespace std;**

**class publication // declaring class Publication**

**{**

**private:**

**string title;**

**float price;**

**public:**

**void add()**

**{**

**cout << "\nEnter the Publication information : " << endl;**

**cout << "Enter Title of the Publication : ";**

**cin.ignore();**

**getline(cin, title);**

**cout << "Enter Price of Publication : ";**

**cin >> price;**

**}**

**void display()**

**{**

**cout << "\n--------------------------------------------------";**

**cout << "\nTitle of Publication : " << title;**

**cout << "\nPublication Price : " << price;**

**}**

**};**

**class book : public publication // declaring class book which inherits class publication in public mode.**

**{**

**private:**

**int page\_count;**

**public:**

**void add\_book()**

**{**

**try**

**{**

**add();**

**cout << "Enter Page Count of Book : ";**

**cin >> page\_count;**

**if (page\_count <= 0)**

**{**

**throw page\_count;**

**}**

**}**

**catch(...)**

**{**

**cout << "\nInvalid Page Count!!!";**

**page\_count = 0;**

**}**

**}**

**void display\_book()**

**{**

**display();**

**cout << "\nPage Count : " <<**

**page\_count;**

**cout << "\n--------------------------------------------------\n";**

**}**

**};**

**class tape : public publication // declaring class tape which inherits class publication in public mode**

**{**

**private:**

**float play\_time;**

**public:**

**void add\_tape()**

**{**

**try**

**{**

**add();**

**cout << "Enter Play Duration of the Tape : ";**

**cin >> play\_time;**

**if (play\_time <= 0)**

**throw play\_time;**

**}**

**catch(...)**

**{**

**cout << "\nInvalid Play Time!!!";**

**play\_time = 0;**

**}**

**}**

**void display\_tape()**

**{**

**display();**

**cout << "\nPlay Time : " <<**

**play\_time << " min";**

**cout << "\n--------------------------------------------------\n";**

**}**

**};**

**int main()**

**{**

**book b1[10]; // object of class book**

**tape t1[10]; // object of class tape**

**int ch, b\_count = 0, t\_count = 0;**

**do**

**{**

**cout << "\n\* \* \* \* \* PUBLICATION DATABASE SYSTEM \* \* \* \* \*";**

**cout << "\n--------------------MENU-----------------------";**

**cout << "\n1. Add Information to Books";**

**cout << "\n2. Add Information to Tapes";**

**cout << "\n3. Display Books Information";**

**cout << "\n4. Display Tapes Information";**

**cout << "\n5. Exit";**

**cout << "\n\nEnter your choice : ";**

**cin >> ch;**

**switch(ch)**

**{**

**case 1:**

**b1[b\_count].add\_book();**

**b\_count + +;**

**break;**

**case 2:**

**t1[t\_count].add\_tape();**

**t\_count + +;**

**break;**

**case 3:**

**cout << "\n\* \* \* \* BOOK PUBLICATION DATABASE SYSTEM \* \* \* \*";**

**for (int j=0;j < b\_count;j++)**

**{**

**b1[j].display\_book();**

**}**

**break;**

**case 4:**

**cout << "\n\* \* \* \* TAPE PUBLICATION DATABASE SYSTEM \* \* \* \*";**

**for (int j=0;j < t\_count;j++)**

**{**

**t1[j].display\_tape();**

**}**

**break;**

**case 5:**

**exit(0);**

**}**

**}while (ch != 5);**

**return 0;**

**}**

**Program 4**

**/\***

**Write a C++ program that creates an output file, writes information to it, closes the file, open it again as an input file and read the information from the file.**

**\*/**

**#include<iostream>**

**#include<fstream>**

**using namespace std;**

**class Employee // declaring class employee**

**{**

**string Name;**

**int ID;**

**double salary;**

**public:**

**void accept()**

**{**

**cout<<"\n Name : ";**

**cin.ignore();**

**getline(cin,Name);**

**cout<<"\n Id : ";**

**cin>>ID;**

**cout<<"\n Salary : ";**

**cin>>salary;**

**}**

**void display()**

**{**

**cout<<"\n Name : "<<Name;**

**cout<<"\n Id : "<<ID;**

**cout<<"\n Salary : "<<salary<<endl;**

**}**

**};**

**int main()**

**{**

**Employee o[5];**

**fstream f;**

**int i,n;**

**f.open("demo.txt",ios::out);**

**cout<<"\n Enter the number of employees you want to store : ";**

**cin>>n;**

**for(i=0;i<n;i++)**

**{**

**cout<<"\n Enter information of Employee "<<i+1<<"\n";**

**o[i].accept();**

**f.write((char\*)&o[i],sizeof o[i]);**

**}**

**f.close();**

**f.open("demo.txt",ios::in);**

**cout<<"\n Information of Employees is as follows : \n";**

**for(i=0;i<n;i++)**

**{**

**cout<<"\nEmployee "<<i+1<<"\n";**

**f.write((char\*)&o[i],sizeof o[i]);**

**o[i].display();**

**}**

**f.close();**

**return 0;**

**}**

**Program 5**

**/\***

**Write a function template for selection sort that inputs, sorts and outputs an integer array and**

**a float array.**

**\*/**

**#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-1;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;**

**int ch;**

**do**

**{**

**cout<<"\n\* \* \* \* \* SELECTION SORT SYSTEM \* \* \* \* \*";**

**cout<<"\n--------------------MENU-----------------------";**

**cout<<"\n1. Integer Values";**

**cout<<"\n2. Float Values";**

**cout<<"\n3. Exit";**

**cout<<"\n\nEnter your choice : ";**

**cin>>ch;**

**switch(ch)**

**{**

**case 1:**

**cout<<"\nEnter total no of int elements:";**

**cin>>n;**

**cout<<"\nEnter int elements:";**

**for(i=0;i<n;i++)**

**{**

**cin>>A[i];**

**}**

**sel(A);**

**break;**

**case 2:**

**cout<<"\nEnter total no of float elements:";**

**cin>>n;**

**cout<<"\nEnter float elements:";**

**for(i=0;i<n;i++)**

**{**

**cin>>B[i];**

**}**

**sel(B);**

**break;**

**case 3:**

**exit(0);**

**}**

**}while(ch!=3);**

**return 0;**

**}**

**Program 6**

**/\***

**Write C++ Program using STL for sorting and searching user defined**

**records such as item records using vector container.**

**\*/**

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

**#include <algorithm> //The STL algorithms 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 // creating class Item**

**{**

**public:**

**char name[10];**

**int quantity;**

**int cost;**

**int code;**

**bool operator==(const Item& i1) //Boolean operators allow**

**you to create more complex conditional statements**

**{**

**if(code==i1.code) //operator will return 1 if the**

**comparison is true, or 0 if the comparison is false**

**return 1;**

**return 0;**

**}**

**bool operator<(const Item& i1)**

**{**

**if(code<i1.code) //operator will return 1 if the**

**comparison is true, or 0 if the comparison 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;**

**cout<<"\n\n";**

**}**

**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!!!";**

**}**

**}**

**Program 7**

#include <iostream>

#include <map>

#include <string>

#include <utility>

using namespace std;

int main()

{

typedef map<string,int> mapType;

mapType populationMap;

populationMap.insert(pair<string, float>("Maharashtra", 125));

populationMap.insert(pair<string, float>("Uttar Pradesh", 225));

populationMap.insert(mapType::value\_type("Bihar", 120));

populationMap.insert(mapType::value\_type("West Bengal", 100));

populationMap.insert(make\_pair("Madhya Pradesh", 90));

populationMap.insert(make\_pair("Tamil Nadu", 80));

populationMap.insert(make\_pair("Rajasthan", 78));

populationMap.insert(make\_pair("Andhra Pradesh", 53));

populationMap.insert(make\_pair("Odisha", 47));

populationMap.insert(make\_pair("Kerala", 38));

populationMap.insert(make\_pair("Telangana", 37));

populationMap.insert(make\_pair("Assam", 35));

populationMap.insert(make\_pair("Jharkhand", 38));

populationMap.insert(make\_pair("Karnataka", 68));

populationMap.insert(make\_pair("Gujarat", 70));

populationMap.insert(make\_pair("Punjab", 31));

populationMap.insert(make\_pair("Chhattisgarh", 30));

populationMap.insert(make\_pair("Haryana", 29));

populationMap.insert(make\_pair("UT Delhi", 19));

populationMap.insert(make\_pair("UT Jammu and Kashmir", 14));

populationMap.insert(make\_pair("Uttarakhand", 12));

populationMap.insert(make\_pair("Himachal Pradesh", 8));

populationMap.insert(make\_pair("Tripura", 04));

populationMap.insert(make\_pair("Meghalaya", 4));

populationMap.insert(make\_pair("Manipur[", 3));

populationMap.insert(make\_pair("Nagaland", 2));

populationMap.insert(make\_pair("Goa", 2));

populationMap.insert(make\_pair("Arunachal Pradesh", 2));

populationMap.insert(make\_pair("UT Puducherry", 2));

populationMap.insert(make\_pair("Mizoram", 1));

populationMap.insert(make\_pair("UT Chandigarh", 1));

populationMap.insert(make\_pair("Sikkim", 1));

populationMap.insert(make\_pair("UT Dadra and Nagar Haveli and Daman and Diu", 1));

populationMap.insert(make\_pair("UT Andaman and Nicobar Islands", 1));

populationMap.insert(make\_pair("UT Lakshadweep", 0.0003));

populationMap.insert(make\_pair("UT Ladakh", 0.00006));

mapType::iterator iter = --populationMap.end();

populationMap.erase(iter);

cout << "Total state and UT of India with Size of populationMap: " << populationMap.size() << '\n';

for (iter = populationMap.begin(); iter != populationMap.end(); ++iter)

{

cout << iter->first <<":" << iter->second << " million\n";

}

char c;

do

{

string state;

cout<<"\nEnter that state you want to know the population of: ";

cin>>state;

iter = populationMap.find(state);

if( iter != populationMap.end() )

cout << state <<"'s populations is "

<< iter->second << " million\n";

else

cout << "State is not in populationMap" << '\n';

cout<<"Do you wish to continue?(y/n):";

cin>>c;

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

populationMap.clear();

return 0;

}