OOP All Practicals ~ By HK_OFFICIAL_

- **Practical 1**

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

1. A constructor (including a default constructor which creates the complex number 0+0i).
2. Overloaded operator + to add two complex numbers.
3. Overloaded operator * to multiply two complex numbers.
4. Overloaded <<and>>to print and read complex numbers.

Program :

```
#include <iostream>

using namespace std;


class complex {

private:

    int real;

    int imag;


public:

    // Default constructor

    complex() {};


    // Member functions

    void setvalue(int, int);
    void display();


    // Operator overloading
    complex operator* (complex &b);
    complex operator+ (complex &d);


    // Friend functions
    friend void operator <<(ostream &output, complex &s);
    friend void operator >>(istream &input, complex &q);
};


// Member function definitions
void complex::setvalue(int c, int d) {
    real = c;
    imag = d;
}


void complex::display() {
    cout << real << "+" << imag << "i" << "\n";
}


// Friend function definitions
void operator<<(ostream &output, complex &s) {
    output << s.real << "+" << s.imag << "i";
```

```
}

void operator>>(istream &input, complex &q) {
    input >> q.real >> q.imag;
}

// Operator overloading definitions
complex complex::operator*(complex &b) {
    complex c7;
    c7.real = real * b.real;
    c7.imag = imag * b.imag;
    return c7;
}

complex complex::operator+(complex &d) {
    complex c8;
    c8.real = real + d.real;
    c8.imag = imag + d.imag;
    return c8;
}

// Main function
int main() {
    complex c1, c2, c3, c4;
    // Input first complex number
    cout << "Enter first No: " << "\n";
    cin >> c3;
    cout << "Object first is:\n" << c3;

    // Input second complex number
    cout << "\nEnter Second No: ";
    cin >> c4;
    cout << "Object second is:" << c4;

    // Perform addition and multiplication
    c1 = c3 + c4;
    c2 = c3 * c4;

    // Output results
    cout << "\n Results are: \n Addition is: " << c1;
    cout << "\n Multiplication is: " << c2;

    return 0;
}
```

- **PRACTICAL 2**

Title-Write a C++ program create a calculator for an arithmetic operator (+, -, *, /). The program should take two operands from user and performs the operation on those two operands depending upon the operator entered by user. Use a switch statement to select the operation. Finally, display the result.

Program :

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

class Calculator {
public:
   float add(double a, double b) {
      return a + b;
   }

   float subtract(double a, double b) {
      return a - b;
   }

   float multiply(double a, double b) {
      return a * b;
   }

   float divide(double a, double b) {
      if (b == 0) {
            cout << "Error: Division by zero." <<endl;

            return 0;

        }

        return a / b;

    }

};


int main() {

    Calculator calc;

    char choice;

    do {

        float num1, num2, result;

        char op;


        cout << "Enter first number: ";

        cin >> num1;

        cout << "Enter second number: ";

        cin >> num2;

        cout << "Enter an operator (+, -, *, /): ";

        cin >> op;


        switch (op) {

            case '+':

                result = calc.add(num1, num2);
                break;

            case '-':

                result = calc.subtract(num1, num2);

                break;

            case '*':

                result = calc.multiply(num1, num2);

                break;

            case '/':

                result = calc.divide(num1, num2);

                break;

            default:

                cout << "Error: Invalid operator." << endl;

                continue;

        }


        cout<< "Result: " << result <<endl;


        cout<< "Do you want to perform another calculation? (y/n): ";

        cin >> choice;


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


    cout << "Thank you for using the calculator!" <<endl;
```

```
    return 0;
}
```

- **PRACTICAL 3**

Title: Develop an object oriented program in C++ to create a database of student information system containing the following information: Name, Roll number, Class, division, Date of Birth, Blood group, Contact address, telephone number, driving license no. etc Construct the database with suitable member functions for initializing and destroying the data viz constructor, default constructor, Copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

Program :

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;

class Student {
    int roll;
    char name[15];
    char address[25];
    char class_name[12];
    char dob[12];
    char licno[15];
    char blood_gr[3];
    char mobile_no[12];
```

```
public:
    // Constructor
    Student() {
        roll = 0;
        strcpy(name, "");
        strcpy(address, "");
        strcpy(class_name, "");
        strcpy(dob, "");
        strcpy(licno, "");
        strcpy(blood_gr, "");
        strcpy(mobile_no, "");
    }

    // Parameterized constructor
    Student(int roll, const char* name, const char* address, const char* class_name, const char*
dob, const char* licno, const char* blood_gr, const char* mobile_no) {
        this->roll = roll;
        strcpy(this->name, name);
        strcpy(this->address, address);
        strcpy(this->class_name, class_name);
        strcpy(this->dob, dob);
        strcpy(this->licno, licno);
        strcpy(this->blood_gr, blood_gr);
        strcpy(this->mobile_no, mobile_no);
}


// Function to accept student details
void accept() {
    cout << "\nEnter roll number: ";
    cin >> roll;
    cout << "Enter name: ";
    cin.ignore();
    cin.getline(name, 15);
    cout << "Enter address: ";
    cin.getline(address, 25);
    cout << "Enter class name: ";
    cin.getline(class_name, 12);
    cout << "Enter DOB (dd/mm/yyyy): ";
    cin.getline(dob, 12);
    cout << "Enter license number: ";
    cin.getline(licno, 15);
    cout << "Enter blood group: ";
    cin.getline(blood_gr, 3);
    cout << "Enter mobile number: ";
    cin.getline(mobile_no, 12);
}


// Function to display student details
```

```
    void display() const {
        cout << "\nRoll: " << roll;
        cout << "\nName: " << name;
        cout << "\nAddress: " << address;
        cout << "\nClass: " << class_name;
        cout << "\nDOB: " << dob;
        cout << "\nLicense No: " << licno;
        cout << "\nBlood Group: " << blood_gr;
        cout << "\nMobile No: " << mobile_no << endl;
    }
};

int main() {
    int ch, n;
    Student students[20];  // Array to store student records

    while (true) {
        cout << "\nMenu\n1. Accept Student Details\n2. Display All Students\n3. Exit\nEnter
Choice: ";
        cin >> ch;
        switch (ch) {
            case 1:
                cout << "Enter number of students: ";
                cin >> n;
                for (int i = 0; i < n; i++) {
```

```
                cout << "\nEntering details for Student " << (i + 1) << ":\n";
                students[i].accept();
            }
            break;

        case 2:
            for (int i = 0; i < n; i++) {
                cout << "\nDisplaying details of Student " << (i + 1) << ":\n";
                students[i].display();
            }
            break;

        case 3:
            exit(0);

        default:
            cout << "\nInvalid choice. Please try again.";
        }
    }

    return 0;
}
```

- **PRACTICAL 4**

Crete User defined exception to check the following conditions and throw the exception if the criterion does not meet.

a. User has age between 18 and 55
b. User stays has income between Rs. 50,000 – Rs. 1,00,000 per month
c. User stays in Pune/ Mumbai/ Bangalore / Chennai
d. User has 4 wheelers

Accept age, Income, City, Vehicle from the user and check for the conditions mentioned

Program :

```
#include <iostream>


using namespace std;


class Details {

int age;

int income;

int vehicle;

string city;


public:

   Details(){

   while (true) {

          try {

             int age;

             cout << "Exception Handling!" << endl;
```

```
            cout << "Enter age: ";
            cin >> age;

            if (age > 18 && age < 55) {
                cout << "Access Granted" << endl;
                break;
            } else {
                throw age;
            }
        } catch (int age) {
            cout << "You are not eligible" << endl;
            cout << "Your Age must be between 18 and 55! \nYour Age is: " << age << endl;
            cout << "Please re-enter your age." << endl;
        }
    }

while(true){

try{
int income;
cout<<"Enter income: ";
cin>>income;
if (income > 50000 && income < 100000 ){
        cout<<"Access Granted"<<endl;
```

```
            break;
        }else{
        throw(income);
        }


    }


    catch(int income){
        cout<<"You are not eligible"<<endl;
        cout<<"Your Income must be between 50000 to 100000! \n Your Income is : "<<income<<endl;
        cout << "Please re-enter your income." << endl;
    }
    }


    while(true){
    try{
    string city;
    cout<<"Enter Your City: ";
    cin>>city;
    if (city == "Pune" || city == "Mumbai" || city == "Bangalore" || city == "Chennai" ){
            cout<<"Access Granted"<<endl;
            break;
        }else{
        throw(city);
        }


    }


    catch(string city){
        cout<<"You are not eligible"<<endl;
        cout<<"You must staying in Cities like -> Pune, Mumbai, Bangalore, Chennai!\n Your City is : "<<city<<endl;
        cout << "Please re-enter your City." << endl;
    }
    }


   while(true){
     try{
    int vehicle;
    cout<<"Enter Your Vehicle Type (Eg: 4 wheeler = 4): ";
    cin>>vehicle;
    if (vehicle == 4){
            cout<<"Access Granted"<<endl;
            break;
        }else{
        throw(vehicle);
        }


    }
    catch(int vehicle){
        cout<<"You are not eligible"<<endl;
        cout<<"You must have 4 wheeler! \n Your Vehicle Type is : "<<vehicle<<" Wheelers"<<endl;
        cout << "Please re-enter your Vehicle Type." << endl;
    }
   }
    }
 };



 int main()
 {
    Details a;
    cout<<"Thankyou for showing your Interest!"<<endl;

    return 0;
 }
```

- **PRACTICAL 5 :**

Title : Write a c++ program tha creates an output file ,writes information toi ,closes the file and open it again as an input file, and read info from file.

Program :

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

class Student {
public:
   int roll;
   char name[30];
   int marks;

   Student(){}

   void getdata();
   void displaydata();
};

void Student :: getdata() {
   cout<< "\nEnter Students Details-->"<<endl;
   cout<< "Enter roll no : ";
   cin >> roll;
   cin.ignore();
   cout<< "Enter Student's name : ";
   cin.getline(name, 30);
```

```
    cout<< "Enter marks : ";

    cin >> marks;

}


void Student :: displaydata(){

    cout<< "\nRoll No : " << roll << endl;

    cout<< "Student's Name : " << name << endl;

    cout<< "Student's marks : " << marks <<endl;

}


int main()

{

    Student s[70];

    fstream file;

    int i,n;


    file.open("C:\\Huzaifa\\xyz.txt",ios::out);


    cout<< "Enter Number of Students : ";

    cin>> n;




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

    s[i].getdata();
    file.write((char *)&s[i], sizeof(s[i]));
    }

    file.close();

    file.open("C:\\Huzaifa\\xyz.txt", ios::in);

    cout<<"\n Reading Student information to the file..."<<endl;

    for(int i=0; i<n; i++){
      file.read((char *)&s[i], sizeof(s[i]));
      s[i].displaydata();
    }

    file.close();
    return 0;
}
```

- **PRACTICAL 6**

Write C++ program using STL for sorting and searching with user defined records such as person record(Name, DOB, Telephone number), Item record (Item code, name, cost,quantity) using vector container.

Program :

```
#include <bits/stdc++.h>
using namespace std;
struct Record {
 string name;
 string dob;
 string telephoneNumber;
};
bool compareRecords(const Record& a, const Record& b) {
 return a.name < b.name;
}
int main() {
 vector<Record>Records;
 int n;
 cout << "Enter the number of personal records: ";
 cin >> n;
 for (int i = 0; i < n; i++) {
 Record record;
 cout << "Enter name: ";
 cin >> record.name;
 cout << "Enter DOB (dd-mm-yyyy): ";
 cin >> record.dob;
 cout << "Enter telephone number: ";
 cin >> record.telephoneNumber;
 Records.push_back(record);
 }
 sort(Records.begin(),Records.end(), compareRecords);
 cout << "Sorted personal records:" << endl;
 for (const Record& record :Records) {
 cout << "Name: " << record.name << ", DOB: " << record.dob << ", Telephone Number: "
<<
record.telephoneNumber << endl;
 }
 string searchName;
 cout << "Enter name to search: ";
 cin >> searchName;
 auto it = find_if(Records.begin(),Records.end(), [&searchName](const Record& record) {
 return record.name == searchName;
 });
 if (it != Records.end()) {
 cout << "Record found: " << it->name << endl;
 } else {
 cout << "Record not found" << endl;
 }
return 0;
}
```

- **PRACTICAL 7**

Write a program in C++ to use map associative container. The keys will be the names of states, and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index, and returns the population of the state.

Program :

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
 string name;
 int pop;
 string states[] = { "Maharashtra", "Bihar", "Uttar Pradesh" };
 int pops[] = { 112374333, 104099452, 199812341 };
 map<string, int> mapStates;
 for (int j = 0; j < 3; j++) {
 name = states[j];
 pop = pops[j];
 mapStates[name] = pop; }
 cout << "Enter state: ";
 cin >> name;
 if (mapStates.find(name) != mapStates.end()) {
 pop = mapStates[name]; // find population
 cout << "Population: " << pop << "\n";
 } else {
 cout << "State not found!\n"; }
 cout << endl;

 for (const auto& iter : mapStates) {

 cout << iter.first << ": " << iter.second << "\n"; }

 return 0;

}
```

In Lab Manual (Extra) : Practical 8 :

Imagine a publishing company which does marketing for book and audio cassette 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, 4 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.

Program :

```
#include <iostream>

#include <string>

using namespace std;


// Base class: Publication

class Publication {

protected:

    string title;

    float price;


public:

    Publication() : title(""), price(0.0) {}
    void getData() {
        cout << "Enter title: ";
        cin.ignore();
        getline(cin, title);
        cout << "Enter price: ";
        cin >> price;

        if (price < 0) {
            throw invalid_argument("Price cannot be negative.");
        }
    }

    void displayData() const {
        cout << "Title: " << title << ", Price: " << price << endl;
    }

    void resetData() {
        title = "";
        price = 0.0;
    }
};

// Derived class: Book
```

```
class Book : public Publication {
private:
   int pageCount;

public:
   Book() : pageCount(0) {}

   void getData() {
      try {
         Publication::getData();
         cout << "Enter page count: ";
         cin >> pageCount;

         if (pageCount < 0) {
            throw invalid_argument("Page count cannot be negative.");
         }
      } catch (exception &e) {
         cout << "Exception: " << e.what() << endl;
         resetData();
      }
   }

   void displayData() const {
      Publication::displayData();
```

```
        cout << "Page Count: " << pageCount << endl;
    }

    void resetData() {
        Publication::resetData();
        pageCount = 0;
    }
};

// Derived class: Tape
class Tape : public Publication {
private:
    float playTime;

public:
    Tape() : playTime(0.0) {}

    void getData() {
        try {
            Publication::getData();
            cout << "Enter play time (in minutes): ";
            cin >> playTime;

            if (playTime < 0) {
                throw invalid_argument("Play time cannot be negative.");
            }
        } catch (exception &e) {
            cout << "Exception: " << e.what() << endl;
            resetData();
        }
    }

    void displayData() const {
        Publication::displayData();
        cout << "Play Time: " << playTime << " minutes" << endl;
    }

    void resetData() {
        Publication::resetData();
        playTime = 0.0;
    }
};

int main() {
    Book book;
    Tape tape;

    cout << "Enter details for book:\n";
    book.getData();


    cout << "\nEnter details for tape:\n";
    tape.getData();


    cout << "\nDisplaying book details:\n";
    book.displayData();


    cout << "\nDisplaying tape details:\n";
    tape.displayData();


    return 0;
}
```

Practical 9 :

A book shop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person inputs the title and author and the system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for 5 the number of copies required. If the requested copies book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed; otherwise the message Required copies not in stock is displayed. Design a system using a class called books with suitable member functions and Constructors. Use new operator in constructors to allocate memory space required. Implement C++ program for the system.

Program :

```
#include <iostream>

#include <string>

using namespace std;


class Book {

private:

    string author;

    string title;

    float price;

    string publisher;

    int stock;


public:

    Book() : author(""), title(""), price(0.0), publisher(""), stock(0) {}


    void getData() {

        cout << "Enter author: ";

        cin.ignore();

        getline(cin, author);

        cout << "Enter title: ";

        getline(cin, title);

        cout << "Enter price: ";

        cin >> price;
        cout << "Enter publisher: ";
        cin.ignore();
        getline(cin, publisher);
        cout << "Enter stock position: ";
        cin >> stock;
    }

    void displayData() const {
        cout << "Author: " << author << "\nTitle: " << title << "\nPrice: " << price
             << "\nPublisher: " << publisher << "\nStock: " << stock << endl;
    }

    bool searchBook(const string& searchTitle, const string& searchAuthor) const {
        return (title == searchTitle && author == searchAuthor);
    }

    bool processSale(int copies) {
        if (copies <= stock) {
            stock -= copies;
            cout << "Total cost: " << (price * copies) << endl;
            return true;
        } else {
            cout << "Required copies not in stock." << endl;
            return false;
        }
    }
};

int main() {
    int numBooks;
    cout << "Enter the number of books: ";
    cin >> numBooks;

    Book* books = new Book[numBooks];

    for (int i = 0; i < numBooks; i++) {
        cout << "\nEnter details for book " << (i + 1) << ":\n";
        books[i].getData();
    }

    string searchTitle, searchAuthor;
    cout << "\nEnter title of the book to search: ";
    cin.ignore();
    getline(cin, searchTitle);
    cout << "Enter author of the book to search: ";
    getline(cin, searchAuthor);

    bool found = false;
    for (int i = 0; i < numBooks; i++) {
        if (books[i].searchBook(searchTitle, searchAuthor)) {
            cout << "\nBook found:\n";
            books[i].displayData();
            found = true;

            int copies;
            cout << "\nEnter the number of copies required: ";
            cin >> copies;
            books[i].processSale(copies);
            break;
        }
    }

    if (!found) {
        cout << "\nBook not found." << endl;
    }

    delete[] books;
    return 0;
}
```

Practical 10 :

Create employee bio-data using following classes i) Personal record ii))Professional record 6 iii)Academic record Assume appropriate data members and member function to accept required data & print bio-data. Create bio-data using multiple inheritance using C++.

Program :

```
#include <iostream>

#include <string>

using namespace std;


// Base class for personal details

class PersonalRecord {

protected:

   string name;

   string dob;

   string address;


public:

   void getPersonalData() {

      cout << "Enter Name: ";

      getline(cin, name);

      cout << "Enter Date of Birth (DD-MM-YYYY): ";

      getline(cin, dob);

      cout << "Enter Address: ";

      getline(cin, address);

   }
```

```
    void displayPersonalData() const {
        cout << "Name: " << name << "\nDate of Birth: " << dob << "\nAddress: " << address
<< endl;
    }
};

// Base class for professional details
class ProfessionalRecord {
protected:
    string company;
    string designation;
    float salary;

public:
    void getProfessionalData() {
        cout << "Enter Company Name: ";
        getline(cin, company);
        cout << "Enter Designation: ";
        getline(cin, designation);
        cout << "Enter Salary: ";
        cin >> salary;
        cin.ignore(); // To ignore the newline character left in the input buffer
    }
    void displayProfessionalData() const {
        cout << "Company: " << company << "\nDesignation: " << designation << "\nSalary: "
<< salary << endl;
    }
};

// Base class for academic details
class AcademicRecord {
protected:
    string highestQualification;
    string university;
    float percentage;

public:
    void getAcademicData() {
        cout << "Enter Highest Qualification: ";
        getline(cin, highestQualification);
        cout << "Enter University Name: ";
        getline(cin, university);
        cout << "Enter Percentage: ";
        cin >> percentage;
        cin.ignore();
    }

    void displayAcademicData() const {
        cout << "Highest Qualification: " << highestQualification << "\nUniversity: " <<
university << "\nPercentage: " << percentage << "%" << endl;
    }
};

// Derived class that combines all records
class BioData : public PersonalRecord, public ProfessionalRecord, public AcademicRecord {
public:
    void getData() {
        cout << "Enter Personal Details:\n";
        getPersonalData();
        cout << "\nEnter Professional Details:\n";
        getProfessionalData();
        cout << "\nEnter Academic Details:\n";
        getAcademicData();
    }

    void displayData() const {
        cout << "\nBio-Data:\n";
        displayPersonalData();
        cout << "\n";
        displayProfessionalData();
        cout << "\n";
        displayAcademicData();
    }
```

```
};

int main() {
   BioData biodata;

   cout << "Create Employee Bio-Data:\n";
   biodata.getData();

   cout << "\nDisplaying Employee Bio-Data:\n";
   biodata.displayData();

   return 0;
}
```

Practical 11 :

Write a function template selection Sort. Write a program that inputs, sorts and outputs an integer array and a float array.

Program :

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

// Function template for selection sort
template <typename T>
```

```
void selectionSort(T arr[], int size) {
    for (int i = 0; i < size - 1; ++i) {
        int minIndex = i;
        for (int j = i + 1; j < size; ++j) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        // Swap the found minimum element with the first element
        T temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}

// Function to display an array
template <typename T>
void displayArray(T arr[], int size) {
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

```
int main() {
    // Integer array
    int intArr[] = {64, 34, 25, 12, 22, 11, 90};
    int intSize = sizeof(intArr) / sizeof(intArr[0]);

    cout << "Original integer array: ";
    displayArray(intArr, intSize);

    selectionSort(intArr, intSize);

    cout << "Sorted integer array: ";
    displayArray(intArr, intSize);

    // Float array
    float floatArr[] = {64.5, 34.2, 25.1, 12.4, 22.9, 11.0, 90.3};
    int floatSize = sizeof(floatArr) / sizeof(floatArr[0]);

    cout << "\nOriginal float array: ";
    displayArray(floatArr, floatSize);

    selectionSort(floatArr, floatSize);

    cout << "Sorted float array: ";
    displayArray(floatArr, floatSize);
    return 0;
}
```