Name : Kunjadiya Brikal Ketanbhai
Roll no : 12
course : MCA
Subject : Object Oriented Concepts and Programming - OOCP

Assignment – 3

1) WAP to create base class Book having int id and char name as data members and respective functionality, show following types of inheritance and display the details of each kind of books, also calculate the total no of each type of books in proper format. Simple inheritance with derived class Sales Hierarchical inheritance with derived classes academics and thrillers Show use of constructor and destructor in above examples of inheritance.

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

class Book
{
protected:
    int id;
    string name;

public:
    Book(int bookId, string bookName) : id(bookId), name(bookName)
    {
        cout << "Book created: " << name << endl;
    }

    virtual void display()
    {
        cout << "Book ID: " << id << ", Book Name: " << name << endl;
    }

    ~Book()
    {
        cout << "Book destroyed: " << name << endl;
    }
};

class Sales : public Book
{
public:
    Sales(int bookId, string bookName) : Book(bookId, bookName) {}

    void display()
    {
        cout << "Sales Book - ";
        Book::display();
    }

    ~Sales()
    {
        cout << "Sales Book destroyed: " << name << endl;
    }
};

class Academics : public Book
{
public:
    Academics(int bookId, string bookName) : Book(bookId, bookName) {}

    void display()
    {
        cout << "Academic Book - ";
        Book::display();
    }

    ~Academics()
    {
        cout << "Academic Book destroyed: " << name << endl;
    }
};

class Thrillers : public Book
{
public:
    Thrillers(int bookId, string bookName) : Book(bookId, bookName) {}

    void display()
    {
        cout << "Thriller Book - ";
        Book::display();
    }

    ~Thrillers()
    {
        cout << "Thriller Book destroyed: " << name << endl;
    }
};

int main()
{
    const int MAX_BOOKS = 100;
    Book *books[MAX_BOOKS];
    int totalSales = 0, totalAcademics = 0, totalThrillers = 0;
    int bookCount = 0;

    int n;
    cout << "Enter the number of books (max " << MAX_BOOKS << "): ";
    cin >> n;

    for (int i = 0; i < n; i++)
    {
        if (bookCount >= MAX_BOOKS)
        {
            cout << "Maximum book limit reached." << endl;
            break;
        }

        int type;
        cout << "Select type of book (1: Sales, 2: Academics, 3: Thrillers): ";
        cin >> type;

        int id;
        string name;
        cout << "Enter Book ID: ";
        cin >> id;
        cout << "Enter Book Name: ";
        cin >> name;
        switch (type)
        {
        case 1:
            books[bookCount] = new Sales(id, name);
            totalSales++;
            break;
        case 2:
            books[bookCount] = new Academics(id, name);
            totalAcademics++;
            break;
        case 3:
            books[bookCount] = new Thrillers(id, name);
            totalThrillers++;
            break;
        default:
            cout << "Invalid type selected. Please enter 1, 2, or 3." << endl;
            i--;
            continue;
        }
        bookCount++;
    }

    cout << "\n--- Book Details ---\n";
    for (int i = 0; i < bookCount; i++)
    {
        books[i]->display();
    }

    cout << "\n--- Total Books ---\n";
    cout << "Total Sales Books: " << totalSales << endl;
    cout << "Total Academic Books: " << totalAcademics << endl;
    cout << "Total Thriller Books: " << totalThrillers << endl;

    for (int i = 0; i < bookCount; i++)
    {
        delete books[i];
    }
    return 0;
}
```

output:-

```
Enter the number of books (max 100): 2
Select type of book (1: Sales, 2: Academics, 3: Thrillers): 1
Enter Book ID: 101
Enter Book Name: advanture
Book created: advanture
Select type of book (1: Sales, 2: Academics, 3: Thrillers): 2
Enter Book ID: 201
Enter Book Name: c++
Book created: c++
--- Book Details ---
Sales Book - Book ID: 101, Book Name: advanture
Academic Book - Book ID: 201, Book Name: c++
--- Total Books ---
Total Sales Books: 1
Total Academic Books: 1
Book destroyed: advanture
Book destroyed: c++
```

2) WAP to create student having data members (rollno, name, stream) as baseclass. Derive class subject with marks of 5 subjects and apply respective functionality. Calculate final result and display details of each student from derived class. (multilevel inheritance).

```
#include <iostream>
using namespace std;
class Student
{
protected:
    int rollno;
    string name;
    string stream;

public:
    void input_student_details()
    {
        cout << "Enter Roll No: ";
        cin >> rollno;
        cout << "Enter Name: ";
        cin >> name;
        cout << "Enter Stream: ";
        cin >> stream;
    }

    void display_student_details()
    {
        cout << "Roll No: " << rollno << endl;
        cout << "Name: " << name << endl;
        cout << "Stream: " << stream << endl;
    }
};

class Subject : public Student
{
protected:
    int marks[5];

public:
    void input_marks()
    {
        cout << "Enter marks for 5 subjects: ";
        for (int i = 0; i < 5; ++i)
        {
            cin >> marks[i];
        }
    }

    int calculate_total_marks()
    {
        int total = 0;
        for (int i = 0; i < 5; ++i)
        {
            total += marks[i];
        }
        return total;
    }
};

class Result : public Subject
{
public:
    void display_details()
    {
        display_student_details();
        cout << "Marks: ";
        for (int i = 0; i < 5; ++i)
        {
            cout << marks[i] << " ";
        }
        cout << endl;
        cout << "Total Marks: " << calculate_total_marks() << endl;
    }
};

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

    Result students[num_students];

    for (int i = 0; i < num_students; ++i)
    {
        cout << "\nEntering details for student " << i + 1 << ":" << endl;
        students[i].input_student_details();
        students[i].input_marks();
    }

    for (int i = 0; i < num_students; ++i)
    {
        cout << "\nDetails of student " << i + 1 << ":" << endl;
        students[i].display_details();
    }
    return 0;
}
```

output:-

```
Enter the number of students: 3
Entering details for student 1:
Enter Roll No: 101
Enter Name: Brikal
Enter Stream: MCA
Enter marks for 5 subjects:
80
59
69
85
90

Entering details for student 2:
Enter Roll No: 102
Enter Name: Mitesh
Enter Stream: MCA
Enter marks for 5 subjects:
75
76
84
81
82

Entering details for student 3:
Enter Roll No: 201
Enter Name: Vansh
Enter Stream: MSC-CS
Enter marks for 5 subjects:
84
82
80
71
69

Details of student 1:
Roll No: 101
Name: Brikal
Stream: MCA
Marks 80 59 69 85 90
Total Marks: 390
Details of student 2:
Roll No: 102
Name: Mitesh
Stream: MCA
Marks: 75 74 84 96 82
Total Marks: 398
Details of student 3:
```

```
Roll No: 201
Name: Vansh
Stream: MCA
Marks: 84 82 80 71 69
Total Marks: 386
```

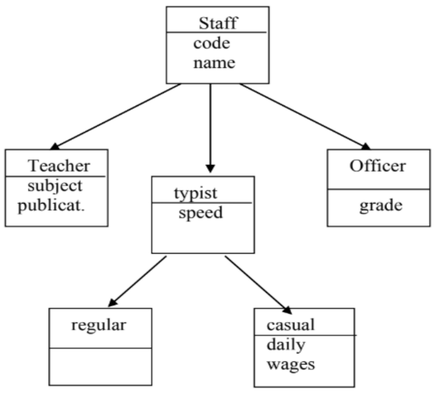
3) WAP to create Class Person, Employee and Department and Apply respective functionalities using inheritance.

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

class Person
{
protected:
    string name;
    int age;

public:
    void setPersonDetails()
    {
        cout << "Enter name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter age: ";
        cin >> age;
    }

    void displayPersonDetails()
    {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};

class Employee : public Person
{
protected:
    int employeeID;
    string position;

public:
    void setEmployeeDetails()
    {
        setPersonDetails();
        cout << "Enter employee ID: ";
        cin >> employeeID;
        cin.ignore(); // Clear the input buffer before getline
        cout << "Enter position: ";
        getline(cin, position);
    }

    void displayEmployeeDetails()
    {
        displayPersonDetails();
        cout << "Employee ID: " << employeeID << endl;
        cout << "Position: " << position << endl;
    }
};

class Department : public Employee
{
    string departmentName;

public:
    void setDepartmentDetails()
    {
        setEmployeeDetails();
        cout << "Enter department name: ";
        getline(cin, departmentName);
    }

    void displayDepartmentDetails()
    {
        displayEmployeeDetails();
        cout << "Department Name: " << departmentName << endl;
    }
};

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

    Department employees[num_emp];

    for (int i = 0; i < num_emp; ++i)
    {
        cout << "\nEntering details for Employee " << i + 1 << ":" << endl;
        employees[i].setDepartmentDetails();
    }

    for (int i = 0; i < num_emp; ++i)
    {
        cout << "\nDetails of Employee " << i + 1 << ":" << endl;
        employees[i].displayDepartmentDetails();
    }

    return 0;
}
```

output:-

```
Enter the number of Employees: 2
Entering details for Employee 1:
Enter name: Brikal
Enter age: 25
Enter employee ID: 101
Enter position: manager
Enter department name: IT
Entering details for Employee 2:
Enter name: Mitesh
Enter age: 26
Enter employee ID: 1032
Enter position: senior devloper
Enter department name: IT
Details of Employee 1:
Name: Brikal
Age: 25
Employee ID: 101
Position: manager
Department Name: IT
Details of Employee 2:
Name: Mitesh
Age: 26
Employee ID: 1032
Position: senior devloper
Department Name: IT
```

4) An educational institution wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown in fig-1. The figure also shows the minimum information required for each class. Specify all the classes and define function to create the database and retrieve individual information as and when required. Write parameterized constructor for each class in the hierarchy.



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

class Staff
{
protected:
    int code;
    string name;

public:
    Staff(int c, const string &n) : code(c), name(n) {}
    virtual void display()
    {
        cout << "Code: " << code << "\nName: " << name << endl;
    }
};

class Teacher : public Staff
{
    string subject;
    int publications;

public:
    Teacher(int c, const string &n, const string &subj, int pub)
        : Staff(c, n), subject(subj), publications(pub) {}

    void display() override
    {
        Staff::display();
        cout << "Subject: " << subject << "\nPublications: " << publications << endl;
    }
};

class Typist : public Staff
{
protected:
    int speed;

public:
    Typist(int c, const string &n, int spd) : Staff(c, n), speed(spd) {}
    virtual void display()
    {
        Staff::display();
        cout << "Speed: " << speed << " words per minute" << endl;
    }
};

class RegularTypist : public Typist
{
public:
    RegularTypist(int c, const string &n, int spd) : Typist(c, n, spd) {}

    void display() override
    {
        Typist::display();
        cout << "Typist Type: Regular" << endl;
    }
};

class CasualTypist : public Typist
{
    int dailyWages;

public:
    CasualTypist(int c, const string &n, int spd, int wages)
        : Typist(c, n, spd), dailyWages(wages) {}
    void display() override
    {
        Typist::display();
        cout << "Typist Type: Casual\nDaily Wages: " << dailyWages << endl;
    }
};

class Officer : public Staff
{
    char grade;

public:
    Officer(int c, const string &n, char g) : Staff(c, n), grade(g) {}

    void display() override
    {
        Staff::display();
        cout << "Grade: " << grade << endl;
    }
};

void displayMenu()
{
    cout << "\nMenu:\n";
    cout << "1. Add Teacher\n";
    cout << "2. Add Regular Typist\n";
    cout << "3. Add Casual Typist\n";
    cout << "4. Add Officer\n";
    cout << "5. Display All Records\n";
    cout << "6. Exit\n";
}

int main()
{
    const int MAX_RECORDS = 10;
    Staff *records[MAX_RECORDS];
    int numRecords = 0;
    int choice;

    while (true)
    {
        displayMenu();
        cout << "Enter your choice: ";
        cin >> choice;

        if (choice == 6)
            break;

        if (numRecords >= MAX_RECORDS)
        {
            cout << "Database is full! Cannot add more records.\n";
            break;
        }
```

```
int code;
string name;

switch (choice)
{
case 1: // Add Teacher
    {
        string subject;
        int publications;
        cout << "Enter Code: ";
        cin >> code;
        cout << "Enter Name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter Subject: ";
        getline(cin, subject);
        cout << "Enter Number of Publications: ";
        cin >> publications;

        records[numRecords++] = new Teacher(code, name, subject, publications);
    }
    break;

case 2: // Add Regular Typist
    {
        int speed;
        cout << "Enter Code: ";
        cin >> code;
        cout << "Enter Name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter Typing Speed (in words per minute): ";
        cin >> speed;

        records[numRecords++] = new RegularTypist(code, name, speed);
    }
    break;

case 3: // Add Casual Typist
    {
        int speed, dailyWages;
        cout << "Enter Code: ";
        cin >> code;
        cout << "Enter Name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter Typing Speed (in words per minute): ";
        cin >> speed;
        cout << "Enter Daily Wages: ";
        cin >> dailyWages;

        records[numRecords++] = new CasualTypist(code, name, speed, dailyWages);
    }
    break;
```

```
        case 4: // Add Officer
            {
                char grade;
                cout << "Enter Code: ";
                cin >> code;
                cout << "Enter Name: ";
                cin.ignore();
                getline(cin, name);
                cout << "Enter Grade: ";
                cin >> grade;

                records[numRecords++] = new Officer(code, name, grade);
            }
            break;

        case 5: // Display All Records
            for (int i = 0; i < numRecords; ++i)
            {
                cout << "\nRecord " << i + 1 << ":\n";
                records[i]->display();
            }
            break;

        default:
            cout << "Invalid choice! Please try again.\n";
        }
    }

    // Clean up memory
    for (int i = 0; i < numRecords; ++i)
    {
        delete records[i];
    }

    return 0;
}
```

output:-

```
Menu:
1. Add Teacher
2. Add Regular Typist
3. Add Casual Typist
4. Add Officer
5. Display All Records
6. Exit
Enter your choice: 4
Enter Code: 401
Enter Name: Brikal
Enter Grade: a

Menu:
1. Add Teacher
2. Add Regular Typist
3. Add Casual Typist
4. Add Officer
```

```
5. Display All Records
6. Exit
Enter your choice: 1
Enter Code: 101
Enter Name: Mitesh
Enter Subject: dbms
Enter Number of Publications: 4

Menu:
1. Add Teacher
2. Add Regular Typist
3. Add Casual Typist
4. Add Officer
5. Display All Records
6. Exit
Enter your choice: 2
Enter Code: 201
Enter Name: Vansh
Enter Typing Speed (in words per minute): 51

Menu:
1. Add Teacher
2. Add Regular Typist
3. Add Casual Typist
4. Add Officer
5. Display All Records
6. Exit
Enter your choice: 5

Record 1:
Code: 401
Name: Brikal
Grade: a

Record 2:
Code: 101
Name: Mitesh
Subject: dbms
Publications: 4

Record 3:
Code: 201
Name: Vansh
Speed: 51 words per minute
Typist Type: Regula
```

---

5) Create a class student from which the classes test and sports are derived. The class student has the name and rollno of the student. The class test has the marks of the internal test and the sports class has the marks of the sports test. The class student contains a virtual function display() which are implemented in the classes test and sports. Write a program which will take relative information and display it using pointer of the base class.

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

class Student
{
protected:
    string name;
    int rollNo;

public:
    void inputStudentDetails()
    {
        cout << "Enter Name: ";
        cin >> name;
        cout << "Enter Roll No: ";
        cin >> rollNo;
    }

    virtual void display()
    {
        cout << "Name: " << name << endl;
        cout << "Roll No: " << rollNo << endl;
    }
};
class Test : public Student
{
    float internalMarks;

public:
    void inputTestMarks()
    {
        cout << "Enter Internal Test Marks: ";
        cin >> internalMarks;
    }

    void display()
    {
        Student::display();
        cout << "Internal Test Marks: " << internalMarks << endl;
    }
};
class Sports : public Student
{
    float sportsMarks;

public:
    void inputSportsMarks()
    {
        cout << "Enter Sports Test Marks: ";
        cin >> sportsMarks;
    }

    void display()
    {
        Student::display();
        cout << "Sports Test Marks: " << sportsMarks << endl;
    }
};

int main()
{
    int choice, numRecords;
    cout << "Enter the number of records you want to insert: ";
    cin >> numRecords;

    for (int i = 0; i < numRecords; ++i)
    {
        cout << "\nEnter the type of student (1-Test, 2-Sports): ";
        cin >> choice;

        if (choice == 1)
        {
            Test testStudent;
            testStudent.inputStudentDetails();
            testStudent.inputTestMarks();
            testStudent.display();
        }
        else if (choice == 2)
        {
            Sports sportsStudent;
            sportsStudent.inputStudentDetails();
            sportsStudent.inputSportsMarks();
            sportsStudent.display();
        }
        else
        {
            cout << "Invalid choice. Please enter 1 for Test or 2 for Sports." << endl;
        }
    }
    return 0;
}
```
----

output:-

```
Enter the number of records you want to insert: 3

Enter the type of student (1-Test, 2-Sports): 1
Enter Name: Brikal
Enter Roll No: 101
Enter Internal Test Marks: 78
Name: Brikal
Roll No: 101
Internal Test Marks: 78

Enter the type of student (1-Test, 2-Sports): 2
Enter Name: Mitesh
Enter Roll No: 201
Enter Sports Test Marks: 68
Name: Mitesh
Roll No: 201
```

```
Sports Test Marks: 68

Enter the type of student (1-Test, 2-Sports): 1
Enter Name: Ram
Enter Roll No: 102
Enter Internal Test Marks: 79
Name: Ram
Roll No: 102
Internal Test Marks: 79
```

---

6) Define a class Figure and use inheritance to define the classes Triangle, Square, Circle, and Rectangle. Provide a container class Frame to contain any number of figures, possibly overlapped in position. Provide operations for drawing frames on the screen and for inserting, modifying, and deleting frame and the contents of frames. (Actual drawing is not required).

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

class Figure
{
public:
    virtual void draw() = 0;
    virtual void modify() = 0;
    virtual void deleteFigure() = 0;
};

class Triangle : public Figure
{
    int base, height;

public:
    Triangle(int b, int h) : base(b), height(h) {}

    void draw()
    {
        cout << "Drawing a triangle with base " << base << " and height " << height << endl;
    }

    void modify()
    {
        cout << "Enter new base and height for the triangle: ";
        cin >> base >> height;
        cout << "Triangle modified." << endl;
    }

    void deleteFigure()
    {
        cout << "Deleting the triangle..." << endl;
    }
};
```

```
class Square : public Figure
{
    int side;
public:
    Square(int s) : side(s) {}

    void draw()
    {
        cout << "Drawing a square with side " << side << endl;
    }
    void modify()
    {
        cout << "Enter new side for the square: ";
        cin >> side;
        cout << "Square modified." << endl;
    }

    void deleteFigure()
    {
        cout << "Deleting the square..." << endl;
    }
};

class Circle : public Figure
{
    int radius;

public:
    Circle(int r) : radius(r) {}
    void draw()
    {
        cout << "Drawing a circle with radius " << radius << endl;
    }
    void modify()
    {
        cout << "Enter new radius for the circle: ";
        cin >> radius;
        cout << "Circle modified." << endl;
    }
    void deleteFigure()
    {
        cout << "Deleting the circle..." << endl;
    }
};
class Rectangle : public Figure
{
    int width, height;

public:
    Rectangle(int w, int h) : width(w), height(h) {}

    void draw()
    {
        cout << "Drawing a rectangle with width " << width << " and height " << height << endl;
    }
    void modify()
    {
        cout << "Enter new width and height for the rectangle: ";
        cin >> width >> height;
        cout << "Rectangle modified." << endl;
    }

    void deleteFigure()
    {
        cout << "Deleting the rectangle..." << endl;
    }
};
class Frame
{
    Figure *figures[10];
    int count;

public:
    Frame() : count(0) {}

    void addFigure(Figure *figure)
    {
        if (count < 10)
        {
            figures[count++] = figure;
        }
        else
        {
            cout << "Frame is full. Cannot add more figures." << endl;
        }
    }
    void drawFrame()
    {
        cout << "Drawing the frame..." << endl;
        for (int i = 0; i < count; i++)
        {
            figures[i]->draw();
        }
    }

    void modifyFigure(int index)
    {
        if (index >= 0 && index < count)
        {
            figures[index]->modify();
        }
        else
        {
            cout << "Invalid index." << endl;
        }
    }
    void deleteFigure(int index)
    {
        if (index >= 0 && index < count)
        {
            figures[index]->deleteFigure();
            for (int i = index; i < count - 1; i++)
            {
                figures[i] = figures[i + 1];
            }
            count--;
        }
        else
        {
            cout << "Invalid index." << endl;
        }
    }
};

int main()
{
    Frame frame;
    int choice;

    while (true)
    {
        cout << "\n1. Add Figure\n2. Draw Frame\n3. Modify Figure\n4. Delete Figure\n5. Exit\nEnter 
your choice: ";
        cin >> choice;

        if (choice == 1)
        {
            int figType;
            cout << "Enter type of figure (1-Triangle, 2-Square, 3-Circle, 4-Rectangle): ";
            cin >> figType;

            if (figType == 1)
            {
                int base, height;
                cout << "Enter base and height of the triangle: ";
                cin >> base >> height;
                frame.addFigure(new Triangle(base, height));
            }
            else if (figType == 2)
            {
                int side;
                cout << "Enter side of the square: ";
                cin >> side;
                frame.addFigure(new Square(side));
            }
            else if (figType == 3)
            {
                int radius;
                cout << "Enter radius of the circle: ";
                cin >> radius;
                frame.addFigure(new Circle(radius));
            }
            else if (figType == 4)
            {
                int width, height;
                cout << "Enter width and height of the rectangle: ";
                cin >> width >> height;
                frame.addFigure(new Rectangle(width, height));
            }
            else
            {
                cout << "Invalid figure type." << endl;
            }
        }
        else if (choice == 2)
        {
            frame.drawFrame();
        }
        else if (choice == 3)
        {
            int index;
            cout << "Enter index of figure to modify: ";
            cin >> index;
            frame.modifyFigure(index);
        }
        else if (choice == 4)
        {
            int index;
            cout << "Enter index of figure to delete: ";
            cin >> index;
            frame.deleteFigure(index);
        }
        else if (choice == 5)
        {
            break;
        }
        else
        {
            cout << "Invalid choice. Please try again." << endl;
        }
    }

    return 0;
}
```
output:-

```
1. Add Figure
2. Draw Frame
3. Modify Figure
4. Delete Figure
5. Exit
Enter your choice: 1
Enter type of figure (1-Triangle, 2-Square, 3-Circle, 4-Rectangle): 2
Enter side of the square: 15

1. Add Figure
2. Draw Frame
3. Modify Figure
4. Delete Figure
5. Exit
```

```
Enter your choice: 1
Enter type of figure (1-Triangle, 2-Square, 3-Circle, 4-Rectangle): 1
Enter base and height of the triangle: 15 30

1. Add Figure
2. Draw Frame
3. Modify Figure
4. Delete Figure
5. Exit
Enter your choice: 1
Enter type of figure (1-Triangle, 2-Square, 3-Circle, 4-Rectangle): 3
Enter radius of the circle: 8

1. Add Figure
2. Draw Frame
3. Modify Figure
4. Delete Figure
5. Exit
Enter your choice: 2
Drawing the frame...
Drawing a square with side 12
Drawing a triangle with base 12 and height 30
Drawing a circle with radius 8

1. Add Figure
2. Draw Frame
3. Modify Figure
4. Delete Figure
5. Exit
Enter your choice: 3
Enter index of figure to modify: 1
Enter new base and height for the triangle: 4 10
Triangle modified.

1. Add Figure
2. Draw Frame
3. Modify Figure
4. Delete Figure
5. Exit
Enter your choice: 2
Drawing the frame...
Drawing a square with side 12
Drawing a triangle with base 4 and height 10
Drawing a circle with radius 8

1. Add Figure
2. Draw Frame
3. Modify Figure
4. Delete Figure
5. Exit
Enter your choice: 5
```

7) Define a class Student. Inherit this class into MCAStudent and NonMCAStudent. Also inherit it into Local and NonLocal students. Multiple inherit LocalMCAStudent from Local and MCAStudent. Define five instance of LocalMCAStudent with a constructor, assuming that all classes have a constructor.

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

class Student
{
protected:
    int rollno;
    string name;

public:
    Student(int r, string n) : rollno(r), name(n) {}

    virtual void display()
    {
        cout << "Roll Number: " << rollno << ", Name: " << name << endl;
    }
};

class MCAStudent : virtual public Student
{
public:
    MCAStudent(int r, string n) : Student(r, n) {}

    void display()
    {
        cout << "MCA Student - ";
        Student::display();
    }
};

class NonMCAStudent : virtual public Student
{
public:
    NonMCAStudent(int r, string n) : Student(r, n) {}

    void display()
    {
        cout << "Non-MCA Student - ";
        Student::display();
    }
};

class Local : virtual public Student
{
public:
    Local(int r, string n) : Student(r, n) {}

    void display()
    {
        cout << "Local Student - ";
        Student::display();
    }
};

class NonLocal : virtual public Student
{
public:
    NonLocal(int r, string n) : Student(r, n) {}

    void display()
    {
        cout << "Non-Local Student - ";
        Student::display();
    }
};

class LocalMCAStudent : public Local, public MCAStudent
{
public:
    LocalMCAStudent(int r, string n) : Student(r, n), Local(r, n), MCAStudent(r, n) {}

    void display()
    {
        cout << "Local MCA Student - ";
        Student::display();
    }
};

int main()
{
    const int numStudents = 5;
    LocalMCAStudent *students[numStudents];

    for (int i = 0; i < numStudents; i++)
    {
        int rollno;
        string name;
        cout << "Enter details for student " << i + 1 << ":" << endl;
        cout << "Roll Number: ";
        cin >> rollno;
        cout << "Name: ";
        cin >> name;

        students[i] = new LocalMCAStudent(rollno, name);
    }

    cout << "\n--- Student Details ---\n";
    for (int i = 0; i < numStudents; i++)
    {
        students[i]->display();
    }

    for (int i = 0; i < numStudents; i++)
    {
        delete students[i];
    }
    return 0;
}
```
output:-

```
Enter details for student 1:
Roll Number: 101
Name: Brikal
Enter details for student 2:
Roll Number: 102
Name: Ronak
Enter details for student 3:
Roll Number: 104
Name: Vansh
Enter details for student 4:
Roll Number: 103
Name: Ram
Enter details for student 5:
Roll Number: 105
Name: Kishore

--- Student Details ---
Local MCA Student - Roll Number: 101, Name: Brikal
Local MCA Student - Roll Number: 102, Name: Ronak
Local MCA Student - Roll Number: 104, Name: Vansh
Local MCA Student - Roll Number: 103, Name: Ram
Local MCA Student - Roll Number: 105, Name: Kishore
```

8) Define a class Faculty. It contains the following attributes:

(a) Name of the faculty
(b) Qualification of the faculty
(c) Subjects the faculty can teach
Inherit the Faculty class into a regular faculty, who
(a) is available full time (no consulting time is specified)
(b) teaches at least three subjects
(c) is part of the institute alone
Inherit the Faculty class into visiting faculty, who
(a) is available only on two to three days (consulting time is also to be specified)
(b) teaches a single subject
(c) is part of some other institute
Provide virtual functions for reading and writing class objects of the classes given above.

```
#include <iostream>
using namespace std;
class Faculty
{
protected:
    string name;
    string qualification;
    string subjects;

public:
    virtual void read()
    {
        cout << "Enter Name: ";
        cin.ignore();
        getline(cin, name);

        cout << "Enter Qualification: ";
        getline(cin, qualification);

        cout << "Enter Subjects (comma-separated): ";
        getline(cin, subjects);
    }

    virtual void display() const
    {
        cout << "Name: " << name << endl;
        cout << "Qualification: " << qualification << endl;
        cout << "Subjects: " << subjects << endl;
    }

    virtual ~Faculty() {}
};

class RegularFaculty : public Faculty
{
public:
    void read() override
    {
        Faculty::read();
        cout << "Regular Faculty: Available full time." << endl;
    }

    void display() const override
    {
        cout << "Regular Faculty Details:" << endl;
        Faculty::display();
    }
};

class VisitingFaculty : public Faculty
{
    string availableDays;
    string consultingTime;

public:
    void read() override
    {
        Faculty::read();
        cout << "Enter Available Days (e.g., Mon, Wed, Fri): ";
        getline(cin, availableDays);

        cout << "Enter Consulting Time: ";
        getline(cin, consultingTime);
    }

    void display() const override
    {
        cout << "Visiting Faculty Details:" << endl;
        Faculty::display();
        cout << "Available Days: " << availableDays << endl;
        cout << "Consulting Time: " << consultingTime << endl;
    }
};

void showMenu()
{
    cout << "\nMenu: \n";
    cout << "1. Add Regular Faculty\n";
    cout << "2. Add Visiting Faculty\n";
    cout << "3. Display Faculty Details\n";
    cout << "4. Exit\n";
    cout << "Enter your choice: ";
}

int main()
{
    const int MAX_FACULTIES = 5;
    Faculty *faculties[MAX_FACULTIES];
    int facultyCount = 0;
    int choice;
    bool running = true;

    while (running)
    {
        showMenu();
        cin >> choice;

        switch (choice)
        {
        case 1:
            if (facultyCount < MAX_FACULTIES)
            {
                Faculty *faculty = new RegularFaculty();
                cout << "\nEnter details for Regular Faculty:" << endl;
                faculty->read();
                faculties[facultyCount++] = faculty;
            }
            else
            {
                cout << "Faculty list is full!" << endl;
            }
            break;

        case 2:
            if (facultyCount < MAX_FACULTIES)
            {
                Faculty *faculty = new VisitingFaculty();
                cout << "\nEnter details for Visiting Faculty:" << endl;
                faculty->read();
                faculties[facultyCount++] = faculty;
            }
            else
            {
                cout << "Faculty list is full!" << endl;
            }
            break;

        case 3:
            if (facultyCount == 0)
            {
                cout << "No faculty records to display." << endl;
            }
            else
            {
                cout << "\n--- Faculty Details ---\n";
                for (int i = 0; i < facultyCount; ++i)
                {
                    faculties[i]->display();
                    cout << endl;
                }
            }
            break;

        case 4: // Exit
            running = false;
            break;

        default:
            cout << "Invalid choice. Please try again." << endl;
        }
    }

    for (int i = 0; i < facultyCount; ++i)
    {
        delete faculties[i];
    }
    return 0;
}
```

output:-

```
Menu:
1. Add Regular Faculty
2. Add Visiting Faculty
3. Display Faculty Details
4. Exit
Enter your choice: 1

Enter details for Regular Faculty:
Enter Name: Brikal
Enter Qualification: MCA
Enter Subjects (comma-separated): c++, rdbms
Regular Faculty: Available full time.
```

```
Menu:
1. Add Regular Faculty
2. Add Visiting Faculty
3. Display Faculty Details
4. Exit
Enter your choice: 2

Enter details for Visiting Faculty:
Enter Name: Vansh
Enter Qualification: M.tech
Enter Subjects (comma-separated): c++, java, softwer developer
Enter Available Days (e.g., Mon, Wed, Fri): mon, thurs, fri
Enter Consulting Time: 1-4

Menu:
1. Add Regular Faculty
2. Add Visiting Faculty
3. Display Faculty Details
4. Exit
Enter your choice: 3

--- Faculty Details ---
Regular Faculty Details:
Name: Brikal
Qualification: MCA
Subjects: c++, dbms
Visiting Faculty Details:
Name: Vansh
Qualification: M.tech
Subjects: c++, java, softwer developer
Available Days: mon, thurs, fri
Consulting Time: 1-4
```

9) Create a base class called shape. Use this class to store two double type values that could be used to compute the area of figures. Derive three specific classes calledtriangle, rectangle and circle from the base shape. Add to the base class, a member function get_data( ) to initialize base class data members and another member function display_area( ) to compute and display the area of figures.Make display_area( ) as a virtual function and redefine this function in derived classes to
suit their requirements. Using these three classes design a program that will accept dimensions of a triangle or rectangle interactively and store it in one array . After having read all the input display the area of all the figures whose area has been read in the program. Remember the two values given as input will be treated as lengths of two sides in the case of rectangle and as base and height in case of triangle. In case of circle only one value should be accepted which will be taken as the radius and the default value of the next parameter should be 0.

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

class Shape
```

```
{
protected:
    double val1, val2;

public:
    virtual void get_data()
    {
        cin >> val1 >> val2;
    }

    virtual void display_area() = 0;
    virtual ~Shape() {}
};

class Triangle : public Shape
{
public:
    void get_data() override
    {
        cout << "Enter base and height for Triangle: ";
        cin >> val1 >> val2;
    }

    void display_area() override
    {
        cout << "Triangle Area: " << 0.5 * val1 * val2 << endl;
    }
};

class Rectangle : public Shape
{
public:
    void get_data() override
    {
        cout << "Enter length and width for Rectangle: ";
        cin >> val1 >> val2;
    }

    void display_area() override
    {
        cout << "Rectangle Area: " << val1 * val2 << endl;
    }
};

class Circle : public Shape
{
public:
    void get_data() override
    {
        cout << "Enter radius for Circle: ";
        cin >> val1;
        val2 = 0;
    }

    void display_area() override
    {
        cout << "Circle Area: " << 3.14 * val1 * val1 << endl;
    }
};

void showMenu()
{
    cout << "\nMenu:\n";
    cout << "1. Add Triangle\n";
    cout << "2. Add Rectangle\n";
    cout << "3. Add Circle\n";
    cout << "4. Display Areas of All Shapes\n";
    cout << "5. Exit\n";
    cout << "Enter your choice: ";
}

int main()
{
    const int MAX_SHAPES = 100;
    Shape *shapes[MAX_SHAPES];
    int shapeCount = 0;
    int choice;
    bool running = true;

    while (running)
    {
        showMenu();
        cin >> choice;

        switch (choice)
        {
        case 1:
            if (shapeCount < MAX_SHAPES)
            {
                shapes[shapeCount] = new Triangle();
                shapes[shapeCount]->get_data();
                ++shapeCount;
            }
            else
            {
                cout << "Shape limit reached!" << endl;
            }
            break;

        case 2:
            if (shapeCount < MAX_SHAPES)
            {
                shapes[shapeCount] = new Rectangle();
                shapes[shapeCount]->get_data();
                ++shapeCount;
            }
            else
            {
                cout << "Shape limit reached!" << endl;
            }
            break;

        case 3:
            if (shapeCount < MAX_SHAPES)
            {
                shapes[shapeCount] = new Circle();
                shapes[shapeCount]->get_data();
                ++shapeCount;
            }
            else
            {
                cout << "Shape limit reached!" << endl;
            }
            break;

        case 4:
            if (shapeCount == 0)
            {
                cout << "No shapes to display." << endl;
            }
            else
            {
                cout << "\n--- Displaying Areas ---\n";
                for (int i = 0; i < shapeCount; ++i)
                {
                    shapes[i]->display_area();
                }
            }
            break;

        case 5: // Exit
            running = false;
            break;

        default:
            cout << "Invalid choice, please try again." << endl;
            break;
        }
    }

    for (int i = 0; i < shapeCount; ++i)
    {
        delete shapes[i];
    }
    return 0;
}
```

output:-

```
Menu:
1. Add Triangle
2. Add Rectangle
3. Add Circle
4. Display Areas of All Shapes
5. Exit
Enter your choice: 1
Enter base and height for Triangle: 15 19

Menu:
1. Add Triangle
2. Add Rectangle
3. Add Circle
4. Display Areas of All Shapes
5. Exit
Enter your choice: 3
Enter radius for Circle: 20

Menu:
1. Add Triangle
2. Add Rectangle
3. Add Circle
4. Display Areas of All Shapes
5. Exit
Enter your choice: 2
Enter length and width for Rectangle: 15 10

Menu:
1. Add Triangle
2. Add Rectangle
3. Add Circle
4. Display Areas of All Shapes
5. Exit
Enter your choice: 4

--- Displaying Areas ---
Triangle Area: 99
Circle Area: 827.14
Rectangle Area: 150

Menu:
1. Add Triangle
2. Add Rectangle
3. Add Circle
4. Display Areas of All Shapes
5. Exit
Enter your choice: 5
```

11) ABC publishing company markets both book and audio cassette versions of its work. Create a class called 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 playing time in minutes (type float). Write a main program that reads both book and tape information in one array. When the user has finished entering data for all books and tapes, displays the resulting data for all the books and tapes entered. Also count no of book and cassette entries in the array using runtime identification feature of C++.

```
#include <iostream>

using namespace std;
```

```
class Publication {
protected:
    string title;
    float price;

public:
    Publication(string t, float p) : title(t), price(p) {}

    virtual void display() const {
        cout << "Title: " << title << endl;
        cout << "Price: $" << price << endl;
    }

    virtual ~Publication() {}
};

class Book : public Publication {
    int pageCount;

public:

    Book(string t, float p, int pages) : Publication(t, p), pageCount(pages) {}

    void display() const override {
        Publication::display();
        cout << "Page Count: " << pageCount << endl;
    }
};

class Tape : public Publication {
    float playingTime;

public:

    Tape(string t, float p, float time) : Publication(t, p), playingTime(time) {}


    void display() const override {
        Publication::display();
        cout << "Playing Time: " << playingTime << " minutes" << endl;
    }
};

// Main program
int main() {
    const int MAX_PUBLICATIONS = 10;
    Publication* publications[MAX_PUBLICATIONS];
    int numPublications = 0;
    int bookCount = 0, tapeCount = 0;

    while (true) {
        cout << "\nMenu:\n";
        cout << "1. Add Book\n";
        cout << "2. Add Tape\n";
        cout << "3. Display All Publications\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        int choice;
        cin >> choice;

        if (choice == 1) {

            string title;
            float price;
            int pageCount;

            cout << "Enter Book Title: ";
            cin.ignore();
            getline(cin, title);
            cout << "Enter Book Price: ";
            cin >> price;
            cout << "Enter Number of Pages: ";
            cin >> pageCount;

            publications[numPublications] = new Book(title, price, pageCount);
            numPublications++;
            bookCount++;
        }
        else if (choice == 2) {

            string title;
            float price, playingTime;

            cout << "Enter Tape Title: ";
            cin.ignore();
            getline(cin, title);
            cout << "Enter Tape Price: ";
            cin >> price;
            cout << "Enter Playing Time (in minutes): ";
            cin >> playingTime;

            publications[numPublications] = new Tape(title, price, playingTime);
            numPublications++;
            tapeCount++;
        }
        else if (choice == 3) {

            cout << "\nDisplaying All Publications:\n";
            for (int i = 0; i < numPublications; i++) {
                publications[i]->display();
                cout << endl;
            }
            cout << "Total Books: " << bookCount << endl;
            cout << "Total Tapes: " << tapeCount << endl;
        }
        else if (choice == 4) {

            cout << "Exiting program\n";
            break;
```

```
        }
        else {
            cout << "Invalid choice. Please try again.\n";
        }
    }

    for (int i = 0; i < numPublications; i++) {
        delete publications[i];
    }

    return 0;
}
```

output:-

```
Menu:
1. Add Book
2. Add Tape
3. Display All Publications
4. Exit
Enter your choice: 1
Enter Book Title: Attack on titan
Enter Book Price: 4000
Enter Number of Pages: 1500

Menu:
1. Add Book
2. Add Tape
3. Display All Publications
4. Exit
Enter your choice: 1
Enter Book Title: legends
Enter Book Price: 7000
Enter Number of Pages: 6000

Menu:
1. Add Book
2. Add Tape
3. Display All Publications
4. Exit
Enter your choice: 2
Enter Tape Title: Unicorn
Enter Tape Price: 1500
Enter Playing Time (in minutes): 500

Menu:
1. Add Book
2. Add Tape
3. Display All Publications
4. Exit
Enter your choice: 3

Displaying All Publications:
Title: Attack on titan
Price: $4000
```

```
Page Count: 1500

Title: legends
Price: $7000
Page Count: 6000

Title: Unicorn
Price: $1500
Playing Time: 500 minutes

Total Books: 2
Total Tapes: 1

Menu:
1. Add Book
2. Add Tape
3. Display All Publications
4. Exit
Enter your choice: 4
Exiting program
```
----

12) Design a manipulator to provide the following output specifications for printing float values.
(i) 5 column width
(ii) Right justified
(iii) 2 digits precision
(iv) Filling unused spaces with +

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

ostream& customFloatFormat(ostream& os) {
    os << fixed;
    os << setprecision(2);
    os << setw(5);
    os << setfill('+');
    return os;
}

int main() {
    float value = 3.14159;
    cout<<"without manipulator: "<<value<<endl;
    cout<<"with manipulator: " << customFloatFormat << value << endl;

    return 0;
}
```

output:-
```
without manipulator: 3.141592761
with manipulator: +3.14
```

13) Define a class marksheet. The class should contain a function PrintMarkSheet such that it prints the marksheet of a given student with three subject names and five marks for each subject. Define manipulators for displaying headings and footnotes. The function should display marksheet with respective headings and class. The marks should be aligned under the headings (Use either ios functions or manipulators).

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

ostream& heading(ostream& os) {
   os << setw(15) << "Student Name"
      << setw(10) << "Math"
      << setw(10) << "Science"
      << setw(10) << "English"
      << endl;
   os << "---------------------------------------------" << endl;
   return os;
}

ostream& footnote(ostream& os) {
   os << "---------------------------------------------" << endl;
   os << "End of MarkSheet" << endl;
   return os;
}

class MarkSheet {
private:
   string studentName;
   int marks[3];

public:

   MarkSheet(string name, int mathMarks, int scienceMarks, int englishMarks) {
      studentName = name;
      marks[0] = mathMarks;
      marks[1] = scienceMarks;
      marks[2] = englishMarks;
   }


   void printMarkSheet() const {
      cout << setw(15) << studentName
           << setw(10) << marks[0]
           << setw(10) << marks[1]
           << setw(10) << marks[2]
           << endl;
   }
};

int main() {
   MarkSheet* students[5];
   string name;
   int mathMarks, scienceMarks, englishMarks;
```

```
    for (int i = 0; i < 5; i++) {
        cout << "Enter details for student " << i + 1 << ":\n";
        cout << "Name: ";
        cin >> name;
        cout << "Marks for Math: ";
        cin >> mathMarks;
        cout << "Marks for Science: ";
        cin >> scienceMarks;
        cout << "Marks for English: ";
        cin >> englishMarks;

        students[i] = new MarkSheet(name, mathMarks, scienceMarks, englishMarks);
        cout << endl;
    }


    cout << "\nMarkSheet of Students:\n";
    heading(cout);

    for (int i = 0; i < 5; i++) {
        students[i]->printMarkSheet();
    }
    footnote(cout);
    for (int i = 0; i < 5; i++) {
        delete students[i];
    }
    return 0;
}
```
output:-
```
Enter details for student 1:
Name: Brikal
Marks for Math: 90
Marks for Science: 70
Marks for English: 79

Enter details for student 2:
Name: Mitesh
Marks for Math: 66
Marks for Science: 89
Marks for English: 96

Enter details for student 3:
Name: Vansh
Marks for Math: 90
Marks for Science: 79
Marks for English: 78

Enter details for student 4:
Name: Virat
Marks for Math: 80
Marks for Science: 69
Marks for English: 89
```

```
Enter details for student 5:
Name: Shobit
Marks for Math: 78
Marks for Science: 59
Marks for English: 90

MarkSheet of Students:
Student Name       Math   Science   English
--------------------------------------------------
             Brikal          90           70           79
             Mitesh          66           89           78
             Vansh           90           79           78
             Virat           87           79           89
            Shobit           78            59           90
--------------------------------------------------
End of MarkSheet
```