QUESTION NO 4 :

#include<iostream>

using namespace std;

class vehical

{

public:

float milage;

int price;

};

class car : public vehical

{

public:

int warrenty;

int seat\_capacity;

string fuel;

};

class bike : public vehical

{

public:

int no\_cylinder;

int no\_gear;

int fuel\_tank;

string wheel;

};

class audi : public car

{

public:

string model\_no;

void set(){

cout<<"set milage and price\n";

cin>>milage>>price;

cout<<"set warenty seat capacity and fuel type\n";

cin>>warrenty>>seat\_capacity>>fuel;

cout<<"set model type type\n";

cin>>model\_no;

}

void print(){

cout<<milage<<" "<<price<<" "<<warrenty<<" "<<seat\_capacity<<" "<<fuel<<" "<<model\_no;

}

};

class ford : public car

{

public:

string model\_no;

void set(){

cout<<"set milage and price\n";

cin>>milage>>price;

cout<<"set warenty seat capacity and fuel type\n";

cin>>warrenty>>seat\_capacity>>fuel;

cout<<"set model type type\n";

cin>>model\_no;

}

void print(){

cout<<milage<<" "<<price<<" "<<warrenty<<" "<<seat\_capacity<<" "<<fuel<<" "<<model\_no<<endl;

}

};

class tvs : public bike

{

public:

string model\_no;

void set(){

cout<<"set milage and price\n";

cin>>milage>>price;

cout<<"set no of cylinder, no of gear, fuel tank wheel type\n";

cin>>no\_cylinder>>no\_gear>>fuel\_tank>>wheel;

cout<<"set model type type\n";

cin>>model\_no;

}

void print(){

cout<<milage<<" "<<price<<" "<<no\_cylinder<<" "<<no\_gear<<" "<<fuel\_tank<<" "<<wheel<<" "<<model\_no<<endl;

}

};

class bajaj : public bike

{

public:

string model\_no;

void set(){

cout<<"set milage and price\n";

cin>>milage>>price;

cout<<"set no of cylinder, no of gear, fuel tank wheel type\n";

cin>>no\_cylinder>>no\_gear>>fuel\_tank>>wheel;

cout<<"set model type type\n";

cin>>model\_no;

}

void print(){

cout<<milage<<" "<<price<<" "<<no\_cylinder<<" "<<no\_gear<<" "<<fuel\_tank<<" "<<wheel<<" "<<model\_no<<endl;

}

};

int main()

{

ford a;

tvs b;

a.set();

a.print();

b.set();

b.print();

return 0;

}

QUESTION NO 3 :

#include <iostream>

#include <string>

using namespace std;

// Base class Marks

class Marks {

public:

Marks(string name) : name(name) {

rollNumber = ++nextRollNumber;

}

// Virtual function to be overridden by derived classes

virtual void inputMarks() = 0;

int getRollNumber() const {

return rollNumber;

}

string getName() const {

return name;

}

int getTotalMarks() const {

return totalMarks;

}

private:

static int nextRollNumber;

int rollNumber;

string name;

int totalMarks = 0;

};

int Marks::nextRollNumber = 0;

// Derived class Physics

class Physics : public Marks {

public:

Physics(string name) : Marks(name) {}

void inputMarks() override {

cout << "Enter Physics marks for " << getName() << ": ";

cin >> physicsMarks;

totalMarks += physicsMarks;

}

private:

int physicsMarks;

};

// Derived class Chemistry

class Chemistry : public Marks {

public:

Chemistry(string name) : Marks(name) {}

void inputMarks() override {

cout << "Enter Chemistry marks for " << getName() << ": ";

cin >> chemistryMarks;

totalMarks += chemistryMarks;

}

private:

int chemistryMarks;

};

// Derived class Mathematics

class Mathematics : public Marks {

public:

Mathematics(string name) : Marks(name) {}

void inputMarks() override {

cout << "Enter Mathematics marks for " << getName() << ": ";

cin >> mathMarks;

totalMarks += mathMarks;

}

private:

int mathMarks;

};

int main() {

int numStudents;

cout << "Enter the number of students in the class: ";

cin >> numStudents;

// Create an array of pointers to the base class Marks

Marks\* students[numStudents];

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

string name;

cout << "Enter the name of student " << i + 1 << ": ";

cin.ignore();

getline(cin, name);

students[i] = new Physics(name);

students[i]->inputMarks();

students[i] = new Chemistry(name);

students[i]->inputMarks();

students[i] = new Mathematics(name);

students[i]->inputMarks();

}

// Calculate the average marks of the class

int totalClassMarks = 0;

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

totalClassMarks += students[i]->getTotalMarks();

}

double classAverage = static\_cast<double>(totalClassMarks) / numStudents;

cout << "\nClass Average Marks: " << classAverage << endl;

// Clean up dynamic memory

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

delete students[i];

}

return 0;

}

QUESTION NO 5:

#include <iostream>

using namespace std;

// Base class Shape

class Shape {

public:

void printShape() {

cout << "This is a shape" << endl;

}

};

// Derived class Polygon inheriting from Shape

class Polygon : public Shape {

public:

void printShape() {

cout << "Polygon is a shape" << endl;

}

};

// Derived class Rectangle inheriting from Polygon

class Rectangle : public Polygon {

public:

void printShape() {

cout << "Rectangle is a polygon" << endl;

}

};

// Derived class Triangle inheriting from Polygon

class Triangle : public Polygon {

public:

void printShape() {

cout << "Triangle is a polygon" << endl;

}

};

// Derived class Square inheriting from Rectangle

class Square : public Rectangle {

public:

void printShape() {

cout << "Square is a rectangle" << endl;

}

};

int main() {

Shape shape;

Polygon polygon;

Rectangle rectangle;

Triangle triangle;

Square square;

cout << "Calling function for each class:" << endl;

shape.printShape();

polygon.printShape();

rectangle.printShape();

triangle.printShape();

square.printShape();

return 0;

}