1.WRITE DOWN THE CODE FOR FOLLOWING DIAGRAM USING POLYMORPHISM

#include <iostream>

class Bank {

public:

virtual float calculateInterestRate() = 0;

};

class SBI : public Bank {

public:

float calculateInterestRate() {

return 5.5f;

}

};

class Axis : public Bank {

public:

float calculateInterestRate() {

return 6.0f;

}

};

class ICIC : public Bank {

public:

float calculateInterestRate() {

return 6.5f; // Specify floating-point literal

}

};

class RateOfInterest : public Bank {

public:

float calculateInterestRate() {

return 5.0f; // Specify floating-point literal

}

};

int main() {

SBI sbi;

Axis axis;

ICIC icic;

RateOfInterest rateOfInterest;

Bank\* ptr = 0; // Use 0 instead of nullptr if using older C++ standard

ptr = &sbi;

std::cout << "SBI Interest Rate: " << ptr->calculateInterestRate() << "%" << std::endl;

ptr = &axis;

std::cout << "Axis Interest Rate: " << ptr->calculateInterestRate() << "%" << std::endl;

ptr = &icic;

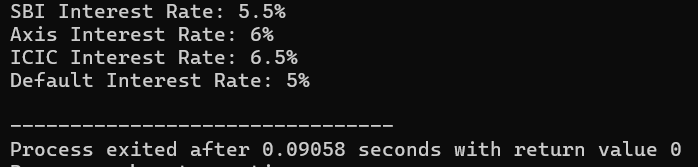
std::cout << "ICIC Interest Rate: " << ptr->calculateInterestRate() << "%" << std::endl;

ptr = &rateOfInterest;

std::cout << "Default Interest Rate: " << ptr->calculateInterestRate() << "%" << std::endl;

return 0;

}



**2.WRITE DOWN SOURCE FOR THE EXCEPTIONS FOR FOLLOWING SNIPPETS, • Separation of Error Handling code from Normal Code • Functions/Methods can handle any exceptions • Notes on Catch all • If an exception is thrown and not caught anywhere, the program terminates abnormally**

#include <iostream>

#include <stdexcept>

using namespace std;

double divide(double a, double b) {

if (b == 0) {

throw runtime\_error("Division by zero error");

}

return a / b;

}

void performCalculation(double a, double b) {

try {

double result = divide(a, b);

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

} catch (const exception& e) {

cerr << "Exception caught: " << e.what() << endl;

}

}

int main() {

double num1, num2;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

performCalculation(num1, num2);

try {

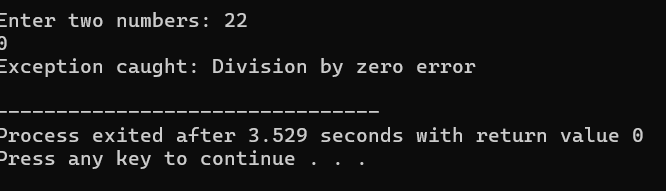
} catch (...) {

cerr << "Unknown exception caught" << endl;

}

return 0;

}



**3. Write down the C++ for Students Report with the implementation of Virtual Function.**

#include <iostream>

#include <string>

using namespace std;

class Student {

protected:

string name;

int rollNumber;

public:

Student(const string& n, int roll) : name(n), rollNumber(roll) {}

virtual void displayDetails() {

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

cout << "Roll Number: " << rollNumber << endl;

}

virtual int calculateTotalMarks() = 0;

};

class UndergraduateStudent : public Student {

protected:

int marksMaths;

int marksScience;

int marksEnglish;

public:

UndergraduateStudent(const string& n, int roll, int math, int sci, int eng)

: Student(n, roll), marksMaths(math), marksScience(sci), marksEnglish(eng) {}

void displayDetails() override {

Student::displayDetails();

cout << "Marks (Maths): " << marksMaths << endl;

cout << "Marks (Science): " << marksScience << endl;

cout << "Marks (English): " << marksEnglish << endl;

}

int calculateTotalMarks() override {

return marksMaths + marksScience + marksEnglish;

}

};

class GraduateStudent : public Student {

protected:

int marksPhysics;

int marksChemistry;

int marksBiology;

public:

GraduateStudent(const string& n, int roll, int phy, int chem, int bio)

: Student(n, roll), marksPhysics(phy), marksChemistry(chem), marksBiology(bio) {}

void displayDetails() override {

Student::displayDetails();

cout << "Marks (Physics): " << marksPhysics << endl;

cout << "Marks (Chemistry): " << marksChemistry << endl;

cout << "Marks (Biology): " << marksBiology << endl;

}

// Override calculateTotalMarks to calculate total marks for graduate students

int calculateTotalMarks() override {

return marksPhysics + marksChemistry + marksBiology;

}

};

int main() {

UndergraduateStudent ugStudent("Alice", 101, 80, 85, 75);

GraduateStudent gradStudent("Bob", 201, 90, 85, 95);

cout << "Undergraduate Student Details:" << endl;

ugStudent.displayDetails();

cout << "Total Marks: " << ugStudent.calculateTotalMarks() << endl;

cout << endl;

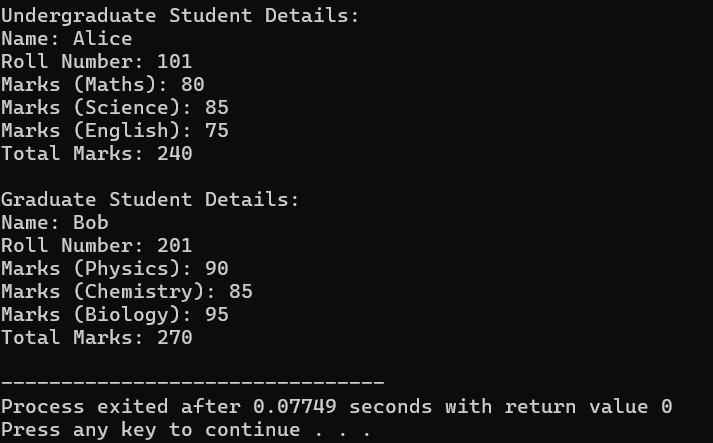
cout << "Graduate Student Details:" << endl;

gradStudent.displayDetails();

cout << "Total Marks: " << gradStudent.calculateTotalMarks() << endl;

return 0;

}

****