**ASSIGNMENT 2**

CO1 : **Apply fundamental OOP concepts to Develop executable programs using the concept of class and objects.**

**1. Write a single C++ program :**

**(i) To find the square root of a number using a function. [Let the return type of the function be void]**

**(ii) To increment a number using an inline function (iii)To decrement a number using an inline function**

**CODE**

#include <iostream>

#include <cmath>

using namespace std;

void findSquareRoot(double num) {

if (num < 0) {

cout << "Cannot find square root of negative numbers." << endl;

return;

}

cout << "Square root of " << num << " is: " << sqrt(num) << endl;

}

inline void increment(int& num) {

num++;

}

inline void decrement(int& num) {

num--;

}

int main() {

double number;

cout << "Enter a number to find its square root: ";

cin >> number;

findSquareRoot(number);

int numToIncrement, numToDecrement;

cout << "Enter a number to increment: ";

cin >> numToIncrement;

increment(numToIncrement);

cout << "After incrementing, the number is: " << numToIncrement << endl;

cout << "Enter a number to decrement: ";

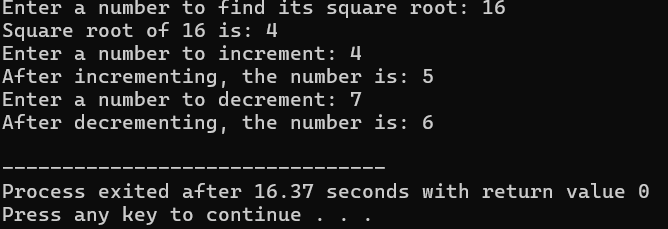
cin >> numToDecrement;

decrement(numToDecrement);

cout << "After decrementing, the number is: " << numToDecrement << endl;

return 0;

}



**2. Create a class student: \* Define a function getdata() and get the name,reg.no and 5 marks of a student. [Note : 1.Use for loop for getting marks,2.Define the function inside the class] \* Define a function tot\_marks() and calculate the total marks. [Note: 1.Use for loop for calculating the total, 2.Define the function outside the class ]**

**CODE**

#include <iostream>

#include <string>

using namespace std;

class Student {

private:

string name;

string regNo;

int marks[5];

public:

void getdata() {

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter student registration number: ";

getline(cin, regNo);

cout << "Enter marks for 5 subjects:" << endl;

for (int i = 0; i < 5; ++i) {

cout << "Enter mark for subject " << i + 1 << ": ";

cin >> marks[i];

}

cin.ignore();

}

int tot\_marks();

};

int Student::tot\_marks() {

int total = 0;

for (int i = 0; i < 5; ++i) {

total += marks[i];

}

return total;

}

int main() {

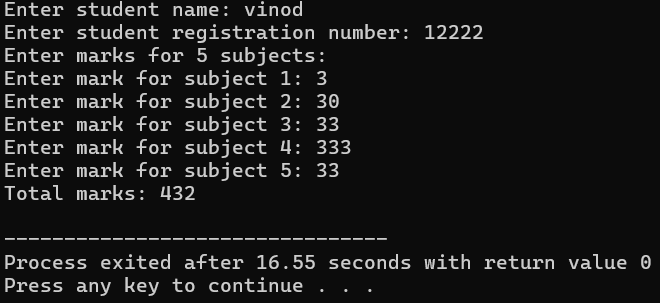
Student student;

student.getdata();

cout << "Total marks: " << student.tot\_marks() << endl;

return 0;

}



**3.Create a class product. \* Define a function get\_product() and get the name of the product and its price. \* Define a function print\_product() and display the product and its price. \* Create an array of object to call the above functions.[Note: Array size: generalized]**

**CODE**

#include <iostream>

#include <string>

using namespace std;

class Product {

private:

string name;

double price;

public:

void get\_product() {

cout << "Enter product name: ";

getline(cin, name);

cout << "Enter product price: ";

cin >> price;

cin.ignore();

}

void print\_product() {

cout << "Product: " << name << ", Price: $" << price << endl;

}

};

int main() {

int n; // Number of products

cout << "Enter the number of products: ";

cin >> n;

cin.ignore();

Product products[n];

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

cout << "Enter details for Product " << i + 1 << ":" << endl;

products[i].get\_product();

}

cout << "\nProduct Details:\n";

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

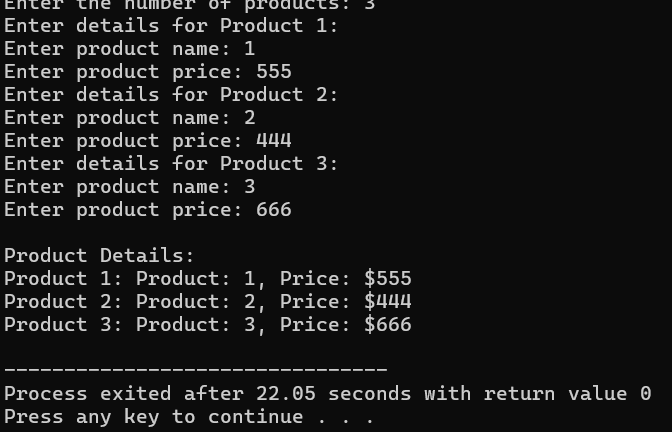
cout << "Product " << i + 1 << ": ";

products[i].print\_product();

}

return 0;

}



**4. Write a C++ program to find the maximum of 2 numbers using a friend function. \* Each number should be got in 2 different classes. \* Define a friend function max which is common to both the class.**

**CODE**

#include <iostream>

using namespace std;

class SecondNumber;

class FirstNumber {

private:

int num1;

public:

FirstNumber(int n) : num1(n) {}

friend int max(const FirstNumber&, const SecondNumber&);

};

class SecondNumber {

private:

int num2;

public:

SecondNumber(int n) : num2(n) {}

friend int max(const FirstNumber&, const SecondNumber&);

};

int max(const FirstNumber& first, const SecondNumber& second) {

return (first.num1 > second.num2) ? first.num1 : second.num2;

}

int main() {

int num1, num2;

cout << "Enter first number: ";

cin >> num1;

cout << "Enter second number: ";

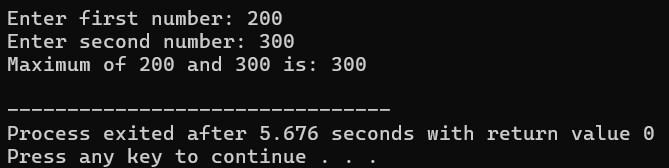
cin >> num2;

int maxNum = max(FirstNumber(num1), SecondNumber(num2));

cout << "Maximum of " << num1 << " and " << num2 << " is: " << maxNum << endl;

return 0;

}



**5.Implement a banking system using C++ classes. Create classes for customers, accounts, and transactions. Apply encapsulation to protect sensitive information, and demonstrate the use of friend functions for access control. Discuss how encapsulation enhances the security and maintainability of the system.**

**CODE**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Account;

class Customer {

private:

string name;

string address;

string phoneNumber;

vector<Account\*> accounts;

public:

Customer(const string& name, const string& address, const string& phoneNumber)

: name(name), address(address), phoneNumber(phoneNumber) {}

void addAccount(Account\* account) {

accounts.push\_back(account);

}

friend class Account;

};

class Account {

private:

int accountNumber;

double balance;

Customer\* owner;

public:

Account(int accountNumber, double balance, Customer\* owner)

: accountNumber(accountNumber), balance(balance), owner(owner) {}

double getBalance() const {

return balance;

}

void deposit(double amount) {

balance += amount;

}

void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

} else {

cout << "Insufficient funds." << endl;

}

}

void transfer(double amount, Account& destination) {

if (amount <= balance) {

withdraw(amount);

destination.deposit(amount);

} else {

cout << "Insufficient funds." << endl;

}

}

void displayInfo() const {

cout << "Account Number: " << accountNumber << endl;

cout << "Owner: " << owner->name << endl;

cout << "Balance: $" << balance << endl;

}

};

class Transaction {

public:

static void displayTransaction(const Account& from, const Account& to, double amount) {

cout << "Transaction Details:" << endl;

cout << "From Account:" << endl;

from.displayInfo();

cout << "To Account:" << endl;

to.displayInfo();

cout << "Amount Transferred: $" << amount << endl;

}

};

int main() {

Customer customer1("John Doe", "123 Main St", "555-1234");

Customer customer2("Jane Smith", "456 Oak St", "555-5678");

Account account1(1001, 5000, &customer1);

Account account2(2002, 3000, &customer2);

customer1.addAccount(&account1);

customer2.addAccount(&account2);

cout << "Initial Account Balances:" << endl;

account1.displayInfo();

account2.displayInfo();

cout << endl;

double transferAmount = 2000;

account1.transfer(transferAmount, account2);

cout << endl << "After Transfer:" << endl;

account1.displayInfo();

account2.displayInfo();

Transaction::displayTransaction(account1, account2, transferAmount);

return 0;

}

