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

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

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

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

#include <iostream>

#include <cmath>

using namespace std;

void findSquareRoot(double num) {

if (num < 0) {

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

return;

}

double squareRoot = sqrt(num);

cout << "Square root of " << num << " is: " << squareRoot << endl;

}

inline void increment(int& num) {

num++;

}

inline void decrement(int& num) {

num--;

}

int main() {

double inputNum;

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

cin >> inputNum;

findSquareRoot(inputNum);

int number;

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

cin >> number;

increment(number);

cout << "After increment: " << number << endl;

decrement(number);

cout << "After decrement: " << number << endl;

return 0;

}

OUTPUT:  
Enter a number to find its square root: 36

Square root of 36 is: 6

Enter a number to increment and decrement: 4

After increment: 5

After decrement: 4

**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 ]**

#include <iostream>

#include <string>

using namespace std;

class Student {

private:

string name;

int regNo;

int marks[5];

public:

void getdata() {

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter registration number: ";

cin >> regNo;

cout << "Enter 5 marks: ";

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

cin >> marks[i];

}

cin.ignore();

}

int tot\_marks() const;

};

int Student::tot\_marks() const {

int total = 0;

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

total += marks[i];

}

return total;

}

int main() {

Student student;

student.getdata();

int totalMarks = student.tot\_marks();

cout << "Total marks: " << totalMarks << endl;

return 0;

}

OUTPUT:

Enter student name: PRIYA

Enter registration number: 192111354

Enter 5 marks: 94

99

91

95

92

Total marks: 471

**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]**

#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() const {

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

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

}

};

int main() {

const int numProducts = 3;

Product products[numProducts];

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

cout << "\nEnter details for Product " << (i + 1) << ":" << endl;

products[i].get\_product();

}

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

cout << "\nDetails for Product " << (i + 1) << ":" << endl;

products[i].print\_product();

}

return 0;

}

OUTPUT:

Enter details for Product 1:

Enter product name: shampoo

Enter product price: 60

Enter details for Product 2:

Enter product name: cookies

Enter product price: 120

Enter details for Product 3:

Enter product name: bangles

Enter product price: 100

Details for Product 1:

Product Name: shampoo

Product Price: $60

Details for Product 2:

Product Name: cookies

Product Price: $120

Details for Product 3:

Product Name: bangles

Product Price: $100

**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.**

#include <iostream>

using namespace std;

class Number1;

class Number2;

int max(const Number1& num1, const Number2& num2);

class Number1 {

private:

int num;

public:

Number1(int n) : num(n) {}

friend int max(const Number1& num1, const Number2& num2);

};

class Number2 {

private:

int num;

public:

Number2(int n) : num(n) {}

friend int max(const Number1& num1, const Number2& num2);

};

int max(const Number1& num1, const Number2& num2) {

return (num1.num > num2.num) ? num1.num : num2.num;

}

int main() {

int num1, num2;

cout << "Enter the first number: ";

cin >> num1;

Number1 n1(num1);

cout << "Enter the second number: ";

cin >> num2;

Number2 n2(num2);

int maximum = max(n1, n2);

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

return 0;

}

OUTPUT:

Enter the first number: 21

Enter the second number: 45

Maximum of 21 and 45 is: 45

**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.**

#include <iostream>

#include <string>

#include <vector>

class Account;

class Transaction;

class Customer {

private:

int customerId;

std::string name;

std::string address;

std::string contactDetails;

public:

Customer(int id, const std::string& n, const std::string& addr, const std::string& contact)

: customerId(id), name(n), address(addr), contactDetails(contact) {}

int getCustomerId() const { return customerId; }

std::string getName() const { return name; }

std::string getAddress() const { return address; }

std::string getContactDetails() const { return contactDetails; }

};

class Account {

private:

int accountNumber;

std::string accountType;

double balance;

public:

Account(int accNum, const std::string& type)

: accountNumber(accNum), accountType(type), balance(0.0) {}

void deposit(double amount) {

balance += amount;

}

bool withdraw(double amount) {

if (balance >= amount) {

balance -= amount;

return true;

}

return false;

}

int getAccountNumber() const { return accountNumber; }

std::string getAccountType() const { return accountType; }

double getBalance() const { return balance; }

};

class Transaction {

private:

int transactionId;

std::string timestamp;

std::string transactionType;

double amount;

public:

Transaction(int id, const std::string& time, const std::string& type, double amt)

: transactionId(id), timestamp(time), transactionType(type), amount(amt) {}

int getTransactionId() const { return transactionId; }

std::string getTimestamp() const { return timestamp; }

std::string getTransactionType() const { return transactionType; }

double getAmount() const { return amount; }

};

int main() {

Customer customer1(1001, "John Doe", "123 Main St", "john.doe@example.com");

Account savingsAccount(2001, "Savings");

Transaction depositTransaction(3001, "2024-03-27 14:30", "Deposit", 1000.0);

savingsAccount.deposit(depositTransaction.getAmount());

std::cout << "Customer: " << customer1.getName() << std::endl;

std::cout << "Account Number: " << savingsAccount.getAccountNumber() << std::endl;

std::cout << "Account Type: " << savingsAccount.getAccountType() << std::endl;

std::cout << "Balance: $" << savingsAccount.getBalance() << std::endl;

return 0;

}

OUTPUT:  
Customer: John Doe

Account Number: 2001

Account Type: Savings

Balance: $1000