**ASSIGNMENT 2**

**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> // for sqrt function

using namespace std;

// Function to find the square root of a number and print the result

void findSquareRoot(double num) {

double result = sqrt(num);

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

}

// Inline function to increment a number

inline void increment(int &num) {

num++;

}

// Inline function to decrement a number

inline void decrement(int &num) {

num--;

}

int main() {

double num;

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

cin >> num;

findSquareRoot(num);

int number;

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

cin >> number;

cout << "Number before increment: " << number << endl;

increment(number);

cout << "Number after increment: " << number << endl;

cout << "Number before decrement: " << number << endl;

decrement(number);

cout << "Number after decrement: " << number << 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**

**Program:-**

#include <iostream>

int main() #include <iostream>

#include <string>

class Student {

private:

std::string name;

int regNo;

int marks[5]; // Assuming 5 subjects

public:

void getData() {

std::cout << "Enter name: ";

std::cin >> name;

std::cout << "Enter registration number: ";

std::cin >> regNo;

std::cout << "Enter marks for 5 subjects:\n";

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

std::cout << "Enter marks for subject " << i + 1 << ": ";

std::cin >> marks[i];

}

}

int totMarks() {

int total = 0;

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

total += marks[i];

}

return total;

}

};

int main() {

Student s;

s.getData();

std::cout << "Total marks: " << s.totMarks() << std::endl;

return 0;

}

// Write C++ code here

std::cout << "Try programiz.pro";

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]**

#include <iostream>

#include <string>

using namespace std;

class Product {

private:

string name;

double price;

public:

void get\_product() {

cout << "Enter product name: ";

getline(cin >> ws, name); // Read the entire line including spaces

cout << "Enter product price: ";

cin >> price;

}

void print\_product() {

cout << "Product Name: " << name << ", Price:RS " << price << endl;

}

};

int main() {

int numProducts;

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

cin >> numProducts;

cin.ignore(); // To consume the newline character left in the buffer

Product products[numProducts];

// Get product details

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

cout << "\nEnter details for product " << i + 1 << ":" << endl;

products[i].get\_product();

}

// Print product details

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

for (int i = 0; i < numProducts; ++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 theh class.**

**Program:-**

#include <iostream>

// Forward declaration of class TwoNumbersB

class TwoNumbersB;

class TwoNumbersA {

private:

int numA;

public:

void getData() {

std::cout << "Enter first number: ";

std::cin >> numA;

}

friend int max(const TwoNumbersA&, const TwoNumbersB&);

};

class TwoNumbersB {

private:

int numB;

public:

void getData() {

std::cout << "Enter second number: ";

std::cin >> numB;

}

friend int max(const TwoNumbersA&, const TwoNumbersB&);

};

int max(const TwoNumbersA& objA, const TwoNumbersB& objB) {

return (objA.numA > objB.numB) ? objA.numA : objB.numB;

}

int main() {

TwoNumbersA a;

TwoNumbersB b;

a.getData();

b.getData();

std::cout << "Maximum of the two numbers: " << max(a, b) << std::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.**

**Program:-**

#include <iostream>

#include <vector>

#include <string>

class Customer {

private:

std::string name;

std::string address;

std::string phoneNumber;

public:

Customer(const std::string& n, const std::string& addr, const std::string& phone)

: name(n), address(addr), phoneNumber(phone) {}

friend class Account;

};

class Account {

private:

std::string accountNumber;

double balance;

Customer\* owner;

public:

Account(const std::string& accNum, double initialBalance, Customer\* cust)

: accountNumber(accNum), balance(initialBalance), owner(cust) {}

void deposit(double amount) {

balance += amount;

}

bool withdraw(double amount) {

if (balance >= amount) {

balance -= amount;

return true;

}

std::cout << "Insufficient funds!" << std::endl;

return false;

}

double gersBalance() const {

return balance;

}

friend class Transaction;

};

class Transaction {

public:

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

if (from.withdraw(amount)) {

to.deposit(amount);

std::cout << "Transfer successful!" << std::endl;

} else {

std::cout << "Transfer failed!" << std::endl;

}

}

};

int main() {

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

Account savingsAccount("SA123456", 1000.0, &customer);

Account checkingAccount("CH789012", 500.0, &customer);

std::cout << "Savings account balance: rs" << savingsAccount.getBalance() << std::endl;

std::cout << "Checking account balance: rs" << checkingAccount.getBalance() << std::endl;

Transaction::transfer(savingsAccount, checkingAccount, 200.0);

std::cout << "Savings account balance: rs" << savingsAccount.getBalance() << std::endl;

std::cout << "Checking account balance: rs" << checkingAccount.getBalance() << std::endl;

return 0;

}