

Object Oriented Programming Lab

Lab Task 6

Submitted by: 241484\_Huzaifa Basit

To: Sir Mahaz

Cyber Security Fall 2024-B

1. Date

C++ Code:

#include<iostream>

using namespace std;

class Date {

int month, day, year;

public:

Date(int m=1, int d=1, int y=2000) {

month = m;

day = d;

year = y;

}

void input\_date();

void print\_format1();

void print\_format2();

void print\_format3();

};

int main() {

Date today;

today.input\_date();

today.print\_format1();

today.print\_format2();

today.print\_format3();

}

void Date::input\_date() {

cout << "Enter month (1-12): ";

cin >> month;

while (month < 1 || month > 12) {

cout << "Invalid month. Enter again: ";

cin >> month;

}

cout << "Enter day (1-31): ";

cin >> day;

while (day < 1 || day > 31) {

cout << "Invalid day. Enter again: ";

cin >> day;

}

cout << "Enter year: ";

cin >> year;

}

void Date::print\_format1() {

cout << month << "/" << day << "/" << year << endl;

}

void Date::print\_format2() {

string month\_names[] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};

cout << month\_names[month - 1] << " " << day << ", " << year << endl;

}

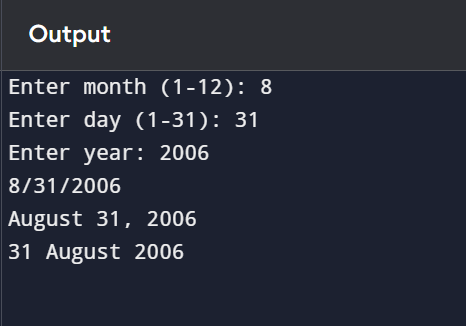
void Date::print\_format3() {

string month\_names[] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};

cout << day << " " << month\_names[month - 1] << " " << year << endl;

}

Output:



1. Student

C++ Code:

#include<iostream>

using namespace std;

class Student {

string name;

int roll\_no;

int marks[5];

public:

Student(string n="", int r=0) {

name = n;

roll\_no = r;

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

marks[i] = 0;

}

}

void input\_details();

void calculate\_grade();

};

int main() {

Student students[10];

for (int i = 0; i < 10; i++) {

students[i].input\_details();

}

for (int i = 0; i < 10; i++) {

students[i].calculate\_grade();

}

}

void Student::input\_details() {

cout << "Enter name: ";

cin >> name;

cout << "Enter roll number: ";

cin >> roll\_no;

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

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

cin >> marks[i];

}

}

void Student::calculate\_grade() {

int total = 0;

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

total += marks[i];

}

float average = total / 5.0;

char grade;

if (average >= 90) grade = 'A';

else if (average >= 80) grade = 'B';

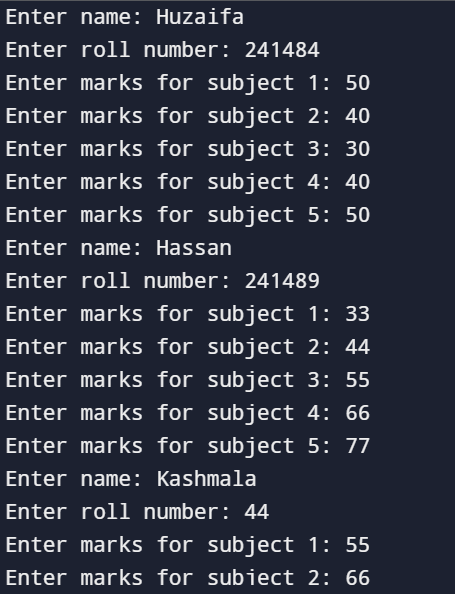
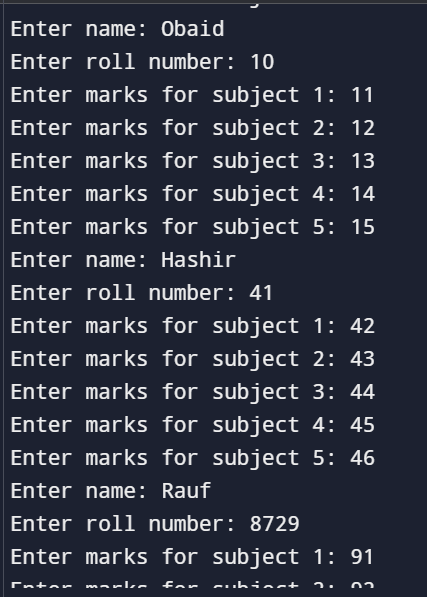
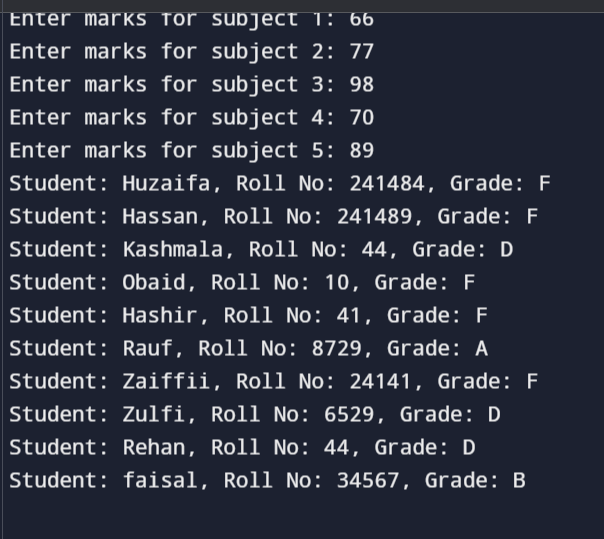
else if (average >= 70) grade = 'C';

else if (average >= 60) grade = 'D';

else grade = 'F';

cout << "Student: " << name << ", Roll No: " << roll\_no << ", Grade: " << grade << endl;

}

Output:  
  
  


1. Person

C++ Code:

#include<iostream>

using namespace std;

class Person {

string name;

int age;

public:

Person(string n="", int a=0) {

name = n;

age = a;

}

void input\_details();

int get\_age() { return age; }

};

int main() {

Person people[20];

int child\_count = 0, teenager\_count = 0, adult\_count = 0, senior\_count = 0;

int child\_age = 0, teenager\_age = 0, adult\_age = 0, senior\_age = 0;

for (int i = 0; i < 20; i++) {

people[i].input\_details();

if (people[i].get\_age() <= 12) {

child\_count++;

child\_age += people[i].get\_age();

} else if (people[i].get\_age() <= 19) {

teenager\_count++;

teenager\_age += people[i].get\_age();

} else if (people[i].get\_age() <= 59) {

adult\_count++;

adult\_age += people[i].get\_age();

} else {

senior\_count++;

senior\_age += people[i].get\_age();

}

}

if (child\_count > 0) cout << "Average age of children: " << (child\_age / child\_count) << endl;

if (teenager\_count > 0) cout << "Average age of teenagers: " << (teenager\_age / teenager\_count) << endl;

if (adult\_count > 0) cout << "Average age of adults: " << (adult\_age / adult\_count) << endl;

if (senior\_count > 0) cout << "Average age of seniors: " << (senior\_age / senior\_count) << endl;

}

void Person::input\_details() {

cout << "Enter name: ";

cin >> name;

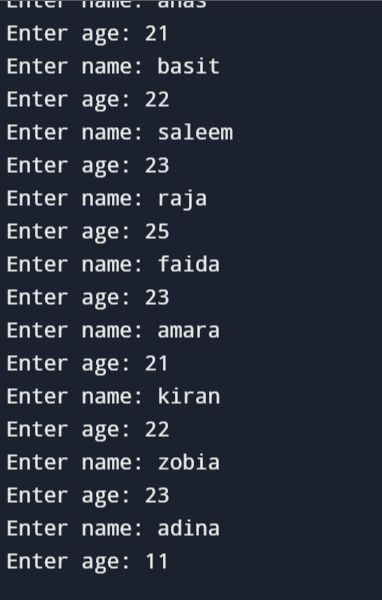
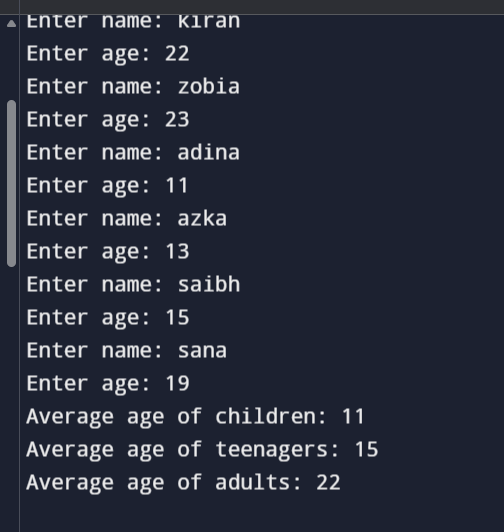
cout << "Enter age: ";

cin >> age;

}

Output:



1. Shape

C++ Code:

#include<iostream>

using namespace std;

class Shape {

string type;

double length, width;

public:

Shape(string t="", double l=0, double w=0) {

type = t;

length = l;

width = w;

}

void input\_shape();

double calculate\_area() { return length \* width; }

};

int main() {

Shape\* shapes[3];

for (int i = 0; i < 3; i++) {

shapes[i] = new Shape();

shapes[i]->input\_shape();

cout << "Area of " << shapes[i]->calculate\_area() << endl;

}

// Cleanup

for (int i = 0; i < 3; i++) {

delete shapes[i];

}

}

void Shape::input\_shape() {

cout << "Enter shape type: ";

cin >> type;

cout << "Enter length: ";

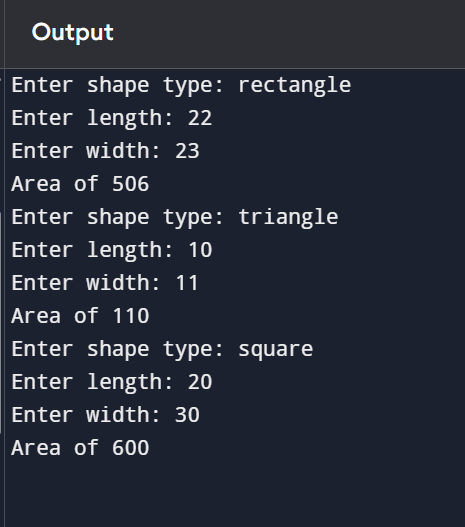
cin >> length;

cout << "Enter width: ";

cin >> width;

}

Output:



1. Update Age

C++ Code:

#include <iostream>

#include <string>

// Person Class Definition

class Person {

public:

std::string name;

int age;

// Constructor

Person(std::string personName, int personAge) : name(personName), age(personAge) {}

// Method to print person details

void printDetails() {

std::cout << "Name: " << name << ", Age: " << age << std::endl;

}

};

// Function to update age

void updateAge(Person\* personPtr, int newAge) {

personPtr->age = newAge; // Update the age

}

// Main Function

int main() {

// Create a Person object

Person Huzaifa("Huzaifa", 18);

// Print initial details

Huzaifa.printDetails();

// Update age

updateAge(&Huzaifa, 20);

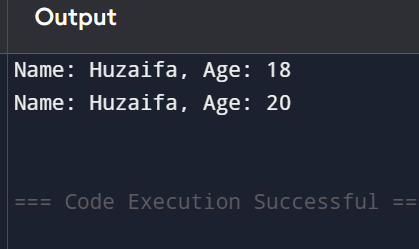
// Print updated details

Huzaifa.printDetails();

return 0;

}

Output:



1. Computer system class hiearchy

C++ Code:

#include <iostream>

#include <string>

class CPU {

// CPU properties can be added here

};

class Memory {

// Memory properties can be added here

};

class Storage {

// Storage properties can be added here

};

class Computer {

private:

CPU cpu;

Memory memory;

Storage storage;

public:

// Computer methods can be added here

void display() {

std::cout << "Computer System with CPU, Memory, and Storage." << std::endl;

}

};

// Main Function

int main() {

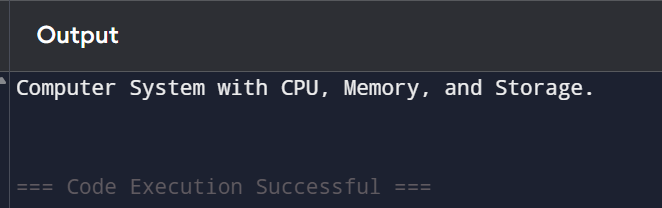
Computer myPC;

myPC.display();

return 0;

}

Output:



1. Autonomous Vehicle simulation

C++ Code:

#include <iostream>

#include <vector>

class Sensor {

// Sensor properties can be added here

};

class Actuator {

// Actuator properties can be added here

};

class Processor {

// Processor properties can be added here

};

class AutonomousVehicle {

private:

std::vector<Sensor> sensors; // Multiple sensors

std::vector<Actuator> actuators; // Multiple actuators

Processor processor; // Central processor

public:

void simulate() {

std::cout << "Autonomous Vehicle is operating with sensors and actuators." << std::endl;

}

};

// Main Function

int main() {

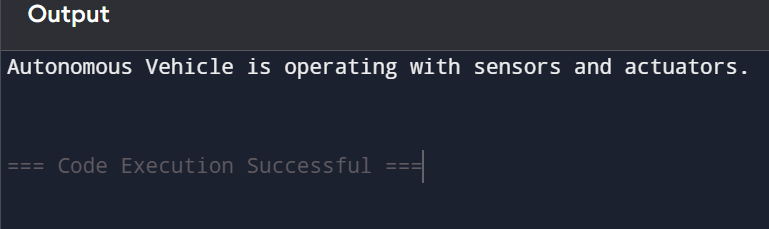
AutonomousVehicle myCar;

myCar.simulate();

return 0;

}

Output:



1. Library management system

C++ Code:

#include <iostream>

#include <vector>

class Book {

// Book properties can be added here

};

class Author {

// Author properties can be added here

};

class Reader {

// Reader properties can be added here

};

class Library {

private:

std::vector<Book> books; // Aggregate multiple books

public:

void addBook(const Book& book) {

books.push\_back(book);

}

};

// Main Function

int main() {

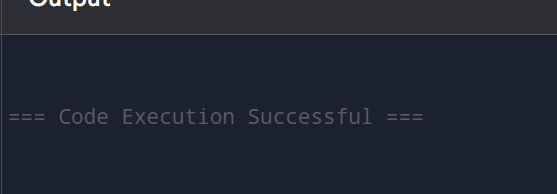
Library myLibrary;

// Example of adding books to library can be added here

return 0;

}

Output:



1. Health Care Management System

C++ Code:

#include <iostream>

#include <vector>

class MedicalRecord {

// MedicalRecord properties can be added here

};

class Physician {

// Physician properties can be added here

};

class Patient {

// Patient properties can be added here

};

class Department {

private:

std::vector<Physician> physicians; // Aggregate physicians

std::vector<Patient> patients; // Aggregate patients

public:

void addPhysician(const Physician& physician) {

physicians.push\_back(physician);

}

};

// Hospital and HealthcareSystem classes can be added here

// Main Function

int main() {

// Implementation of hospital and healthcare system can be added here

return 0;

}

Output:

