OOPS Concept Assignment - by dhavalbhatti

P-1: WAP to print “Hello World” using C++

```cpp
#include <iostream>
using namespace std;
int main()
{
cout << "Hello World" << endl;
return 0;
}
```

P-2: What is OOP? List OOP concepts

C++ uses object-oriented programming (OOP) to organize code like a factory making cars.

OOP Concepts:
1. Class: Blueprint (e.g., car design).
2. Object: Actual car made from the blueprint.
3. Inheritance: A sports car inherits features from a basic car.
4. Polymorphism: A car used as both a taxi & family car.
5. Encapsulation: Keeping the car's engine hidden.
6. Abstraction: Driving without knowing engine details.

P-3: What is the difference between OOP and POP?

OOP (Object-Oriented Programming) focuses on "objects" representing data and methods. It uses classes, inheritance, and encapsulation, making it easy to model real-world things and their interactions. Examples: Java, C++.

POP (Procedure-Oriented Programming) organizes code around functions, with data passed between them. It's more linear and suited for tasks with a clear sequence. Example: C.

In OOP, think "things and actions"; in POP, think "steps to follow."

P-4: WAP to create a simple calculator using class

```cpp
#include <iostream>
using namespace std;

class Calculator
{
public:
double add(double a, double b)
{
return a + b;
}
double subtract(double a, double b)
{
return a - b;
}
double multiply(double a, double b)
{
return a * b;
}
double divide(double a, double b)
{
return a / b;
}
};

int main()
{
Calculator obj;
double a, b;
cout << "Enter two numbers: " << endl;
cin >> a >> b;

cout << "Addition: " << obj.add(a, b) << endl;
cout << "Subtraction: " << obj.subtract(a, b) << endl;
cout << "Multiplication: " << obj.multiply(a, b) << endl;
cout << "Division: " << obj.divide(a, b) << endl;
return 0;
}
```

P-5: Define a class to represent a bank account.
Include the following members:
1. Data Member: -Name of the depositor -Account Number -Type of Account -Balance amount in the account
2. Member Functions -To assign values -To deposited an amount -To withdraw an amount after checking balance -To display name and balance

```
#include <iostream>
using namespace std;

class BankAccount
{
public:
string name;
int account_number;
string account_type;
double balance;
void display()
{
cout << "Name: " << name << endl;
cout << "Account Number: " << account_number << endl;
cout << "Account Type: " << account_type << endl;
cout << "Balance: " << balance << endl;
}
};
int main()
{
BankAccount obj;
obj.name = "Dhaval Bhatti";
obj.account_number = 123456789;
obj.account_type = "Current";
obj.balance = 4000000000.00;
obj.display();
return 0;
}
```

P-6: Write a program to find the multiplication values and the cubic values using the inline function

```
#include <iostream>
using namespace std;

inline int square(int s)
{
return s * s;
}

inline int cube(int s)
{
return s * s * s;
}

int main()
{
int side;
cout << "Enter the value: ";
cin >> side;
cout << "The area of the square is: " << square(side) << endl;
cout << "The volume of the cube is: " << cube(side) << endl;
}
```

P-7: Write a program of Addition, Subtraction, Division, and Multiplication using the constructor.

```
#include <iostream>
using namespace std;

class Add
{
public:
Add(double a, double b)
{
cout << "Addition: " << a + b << endl;
}
};
```

```
class Subtract
{
public:
Subtract(double a, double b)
{
cout << "Subtraction: " << a - b << endl;
}
};

class Multiply
{
public:
Multiply(double a, double b)
{
cout << "Multiplication: " << a * b << endl;
}
};

class Divide
{
public:
Divide(double a, double b)
{
cout << "Division: " << a / b << endl;
}
};

int main()
{
double a, b;
cout << "Enter two numbers: " << endl;
cin >> a >> b;

Add obj1(a, b);
Subtract obj2(a, b);
Multiply obj3(a, b);
Divide obj4(a, b);
return 0;
}
```

P-8: Assume a class cricketer is declared. Declare a derived class batsman from cricketer. Data member of batsman. Total runs, Average runs and best performance. Member functions input data, calculate average runs, Display data. (Single Inheritance)

```
#include <iostream>
using namespace std;

class Cricketer
{
public:
void Totalruns(double a, double b, double c)
{
cout << "Total runs are " << a + b + c << endl;
}
void Averageruns(double a, double b, double c)
{
cout << "Average runs are " << (a + b + c) / 3 << endl;
}
void Bestscore(double a, double b, double c)
{
if (a > b && a > c)
{
cout << "Best score is " << a << endl;
}
else if (b > a && b > c)
{
cout << "Best score is " << b << endl;
}
else
{
cout << "Best score is " << c << endl;
}
}
};

class Batsman : public Cricketer
{
public:
void Match1(double a)
{
cout << "Match 1 runs: " << a << endl;
}
void Match2(double b)
```

```
{
cout << "Match 2 runs: " << b << endl;
}
void Match3(double c)
{
cout << "Match 3 runs: " << c << endl;
}
};

int main()
{
Batsman obj;
double a, b, c;
cout << "Enter runs of match 1: " << endl;
cin >> a;
obj.Match1(a);
cout << "Enter runs of match 2: " << endl;
cin >> b;
obj.Match2(b);
cout << "Enter runs of match 3: " << endl;
cin >> c;
obj.Match3(c);
obj.Totalruns(a, b, c);
obj.Averageruns(a, b, c);
obj.Bestscore(a, b, c);
return 0;
}
```

P-9: Create a class person having members name and age. Derive a class student having member percentage. Derive another class teacher having member salary. Write necessary member function to initialize, read and write data. Write also Main function (Multiple Inheritance)

```cpp
#include <iostream>
using namespace std;

class Person
{
public:
void display(string name,
int age)
{
cout << "Name: " << name << endl;
cout << "Age: " << age << endl;
}
};

class Student : public Person
{
public:
void percentage(double percentage)
{
cout << "Percentage: " << percentage << endl;
}
};

class Teacher : public Person
{
public:
void salary(double salary)
{
cout << "Salary: " << salary << endl;
}
};

int main()
{
Student obj1;
Teacher obj2;
cout << "Student Details: " << endl;
obj1.display("Dhaval Bhatti", 36);
obj1.percentage(99.99);
cout << "Teacher Details: " << endl;
obj2.display("Darshan Gada", 24);
obj2.salary(100000.00);
}
```

P-10: Assume that the test results of a batch of students are stored in three different classes. Class Students are storing the roll number. Class Test stores the marks obtained in two subjects and class result contains the total marks obtained in the test. The class result can inherit the details of the marks obtained in the test and roll number of students. (Multilevel Inheritance)

```
#include <iostream>
using namespace std;

class Student
{

public:
string name;
int roll_no;
void get_student_info()
{
cout << "Enter the name of the student: ";
cin >> name;
cout << "Enter the roll number of the student: ";
cin >> roll_no;
}
void display_student_info()
{
cout << "Name: " << name << endl;
cout << "Roll Number: " << roll_no << endl;
}
};

class Test: public Student
{
public:
int marks1, marks2;
void get_test_marks()
{
cout << "Enter the marks of the first test: ";
cin >> marks1;
cout << "Enter the marks of the second test: ";
cin >> marks2;
}
void display_test_marks()
{
cout << "Marks of the first test: " << marks1 << endl;
```

```
cout << "Marks of the second test: " << marks2 << endl;
}
};

class Result : public Test
{
public:
void display_result()
{
display_student_info();
}
void display_total_marks()
{
cout << "Total marks: " << marks1 + marks2 << endl;
}
};

int main()
{
Result obj;
obj.get_student_info();
obj.get_test_marks();
cout << "Student Scorecard: " << endl;
obj.display_result();
obj.display_test_marks();
obj.display_total_marks();
return 0;
}
```

P-11: Write a program to Mathematic operation like Addition, Subtraction, Multiplication, Division Of two number using different parameters and Function Overloading

```
#include<iostream>
using namespace std;

class Calculator {
public:
void Result(int a, int b) {
cout << "Original | Sum of " << a << " and " << b << " is " << a + b << endl;
cout << "Original | Substraction of " << a << " and " << b << " is " << a - b <<
endl;
cout << "Original | Multiplication of " << a << " and " << b << " is " << a * b <<
endl;
cout << "Original | Division of " << a << " and " << b << " is " << a / b << endl;
}
void Result(double a, double b) {
cout << "Overloading | Sum of " << a << " and " << b << " is " << a + b << endl;
cout << "Overloading | Substraction of " << a << " and " << b << " is " << a - b <<
endl;
cout << "Overloading | Multiplication of " << a << " and " << b << " is " << a * b
<< endl;
cout << "Overloading | Division of " << a << " and " << b << " is " << a / b <<
endl;
}
};

int main(){
Calculator obj;
obj.Result(5,7);
obj.Result(10.3,20.5);
return 0;
}
```

P-12: Write a Program of Two 1D Matrix Addition using Operator Overloading

```
#include <iostream>
using namespace std;

class Matrix
{
public:
int a;
int b;

Matrix(int c, int d)
{
a = c;
b = d;
}
Matrix operator+(Matrix obj)
{
Matrix result(0, 0);
result.a = a + obj.a;
result.b = b + obj.b;
return result;
}

void display()
{
cout << "Matrix addition = " << a << ", " << b << endl;
}
};

int main()
{
Matrix obj1(10, 20);
Matrix obj2(30, 40);
Matrix obj3(0, 0);
obj3 = obj1 + obj2;
obj3.display();
return 0;
}
```

P-13: Write a program to concatenate the two strings using Operator Overloading -

```
#include<iostream>
using namespace std;

class Object{
public:
string a;
string b;

Object(string c, string d){
a = c;
b = d;
}

Object operator+(Object obj){
Object result("","");
result.a = a + obj.a;
result.b = b + obj.b;
return result;
}

void display(){
cout<<"Final string = "<<a<< ", " <<b<<endl;
}
};

int main(){
Object obj1("Hello ","How ");
Object obj2("Dhaval","are you doing?");
Object obj3("","");
obj3 = obj1 + obj2;
obj3.display();
return 0;
}
```

P-14: Write a program to calculate the area of circle, rectangle and triangle using Function Overloading

```
#include<iostream>
using namespace std;

class Area {
public:
void Result(int a, int b) {
cout << "Area of Rectangle: " << a * b << endl;
}
void Result(double a, double b) {
cout << "Area of Triangle: " << 0.5 * a * b << endl;
}
void Result(double a) {
cout << "Area of Circle: " << 3.14 * a * a << endl;
}
};

int main(){
Area obj;
obj.Result(5,7);
obj.Result(10.3,20.5);
obj.Result(6.78);
return 0;
}
```

P-15: Write a program to swap the two numbers using the friend function without using third variable

```
#include <iostream>
using namespace std;
```

```
class Original
{
private:
int a, b;
void get_numbers()
{
cout << "Enter the first number: ";
cin >> a;
cout << "Enter the second number: ";
cin >> b;
}
void display_numbers()
{
cout << "Before swapping: " << endl;
cout << "First number: " << a << endl;
cout << "Second number: " << b << endl;
swap(a, b);
}

void display_swapped_numbers()
{
cout << "After swapping: " << endl;
cout << "First number: " << a << endl;
cout << "Second number: " << b << endl;
}

public:
friend void swapped(Original obj);
};

void swapped(Original obj)
{
obj.get_numbers();
obj.display_numbers();
obj.display_swapped_numbers();
}

int main()
{
Original obj;
swapped(obj);

return 0;
}
```

P-16: Write a program to find the max number from given two numbers using friend function

```
#include <iostream>
using namespace std;

class Max
{
private:
int a, b, c;
void get_numbers()
{
cout << "Enter the first number: ";
cin >> a;
cout << "Enter the second number: ";
cin >> b;
cout << "Enter the third number: ";
cin >> c;
}
void max_number()
{
if (a > b && a > c)
{
cout << "The maximum number is: " << a << endl;
}
else if (b > a && b > c)
{
cout << "The maximum number is: " << b << endl;
}
else
{
cout << "The maximum number is: " << c << endl;
}
}

public:
friend void max(Max obj);
};

void max(Max obj)
{
obj.get_numbers();
obj.max_number();
}
```

```
int main()
{
Max obj;
max(obj);

return 0;
}
```

P-17: Write a program of to swap the two values using templates

```
#include <iostream>
using namespace std;

template <class T>
int swap_numbers(T &x, T &y)
{
T t;
t = x;
x = y;
y = t;
return 0;
}

int main()
{
int a, b;
cout << "Enter the first number: ";
cin >> a;
cout << "Enter the second number: ";
cin >> b;

swap_numbers(a, b);
cout << "After swapping: " << endl;
cout << a << " " << b << endl;
return 0;
}
```

P-18: Write a program of to sort the array using templates.

```cpp
#include <algorithm>
#include <iostream>
using namespace std;

int main()
{
int arr[] = {3, 15, 1, 20, 4};

std::sort(std::begin(arr), std::end(arr));

for (int i : arr)
{
std::cout << i << " ";
}

return 0;
}
```