**Assignment #03**

**Name: Basharat Hussain**

**Roll no: P17-6102**

**Section: A**

**Question #02**

#include<iostream>

using namespace std;

class time {

private:

int hours;

int minutes;

int seconds;

public:

time(){

hours= 0;

minutes= 0;

seconds= 0;

}

time(int h, int m, int s) {

hours= h;

minutes= m;

seconds= s;

}

int getHours(){

return hours;

}

int getMinutes(){

return minutes;

}

int getSeconds() {

return seconds;

}

void display()

{

cout<<hours<<":"<<minutes<<":"<<seconds<<endl;

}

time add(time t1, time t2) {

int hoursAdd = t1.getHours() + t2.getHours();

if (hoursAdd > 23) {

hoursAdd -= 24;

}

int minutesAdd = t1.getMinutes() + t2.getMinutes();

if (minutesAdd > 59) {

minutesAdd -= 60;

hoursAdd += 1;

}

int secondsAdd = t1.getSeconds() + t2.getSeconds();

if (secondsAdd > 59) {

secondsAdd -= 60;

minutesAdd += 1;

}

time t3(hoursAdd, minutesAdd, secondsAdd);

}

};

int main() {

time t1(05, 30, 10);

time t2(07, 20, 10);

time t3;

t3 = t3.add(t1, t2);

cout<<"Time 1 is: ";

t1.display();

cout<<"Time 2 is: ";

t2.display();

cout<<"Addition of two times is: ";

t3.display();

return 0;

}

1. What is the main purpose of using oop ?

Object Oriented Programming The development of complicated computer applications has seen the need for data abstraction with the software engineering field. The main advantages of data abstraction are:

1. It reduces the conceptual load for the programmer.
2. 2. It provides a method of fault containment by preventing the programmer from using code in an inappropriate way.
3. 3. It increases the independence among program components. Programmers can work on individual parts of the code without affecting the interface between components.
4. Can we access the private data member of a class directly in main function c++?if yes how?

Is it possible to access private members outside a class without friend?

Yes, it is possible using pointers. See the following program as an example.

Note that the above way of accessing private data members is not at all a recommended way of accessing members and should never be used. Also, it doesn’t mean that the encapsulation doesn’t work in C++. The idea of making private members is to avoid accidental changes. The above change to data is not accidental. It’s an intentionally written code to fool the compiler

1. How can we count the number of objects created of a class ?

Start with the number zero, and each time an object is created, add one.

That's the logic.

If you mean something like creating a class, and counting how many times an object of that class is created, put the functionality to do the counting inside the constructor and make the counter shared across all instances (for example, by making it static).

1. Why we cant have more than one distructor ?

A **destructor** doesn't **have** parameters, so there can be only **one**. However **you** can **have more than 1** constructor since **you** can overload the constructor which is not possible **with Destructors**.

1. Main difference between class and struct ?

Data Type :

Classes are Reference type and therefore stored on the heapwhile

Structs are Value type and therefore stored on stack. Behaves like simple data types

Inheritance:

classes are Support Inheritance while

Structs are not support inheritance

Default Values

classes are Default value of a class type is null

Sstructs are Default value is the value produced by zeroing out the fields of the struct

Field Initialization:

classes Permits initialization of instance fields while Structs Do not permit initialization of instance fields

Constructors:

Classes Permits declaration of parameter less constructorsWhile Structs are Do not support parameter less constructors

Destructors:

Classes Supportedwhile Structs are Not.