ASSIGNMENT-1

**Ans 1:-**

#include<iostream>

#include<string>

using namespace std;

class automobile

{

public:

int maxSpeed=180;

bool airConditioner;

bool isTheCarOn=true;

string colour;

double weight;

string interior="leather";

int numberOfWheels = 4;

int numberOfHeadlights = 2;

automobile(string clr, bool ac, double wt)

{

colour=clr;

airConditioner=ac;

weight=wt;

}

void display()

{

if(isTheCarOn)

cout<<"The car is working."<<"\n";

else

cout<<"The car is not working."<<"\n";

cout<<"Maximum speed of the car is: "<<maxSpeed<<".\n";

cout<<"It has "<<numberOfWheels<<" wheels.\n";

cout<<"It looks "<<colour<<".\n";

if(airConditioner)

cout<<"The car has air conditioner."<<"\n";

else

cout<<"The car does not have air conditioner."<<"\n";

cout<<"It weighs all of "<<weight<<" kilos.\n";

cout<<"Interiors are "<<interior<<".\n";

}

};

int main()

{

automobile auto1("Red", true, 1670.20), auto2("Lime Green", false, 1800.60);

cout<<"Car 1:\n";

auto1.display();

cout<<"\n\nCar 2:\n\n";

auto2.display();

return 0; }

**Ans 2** **:-**

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling languages in the field of software engineering.

UML offers a way to visualize a system's architectural blueprints in a diagram, including elements such as:

-any activities.

-individual components of the system

and how they can interact with other s/w components.

-how the system will run.

-how entities interact with others.

-external user interface.

Types of UML diagrams:

Class Diagram

Component Diagram

Deployment Diagram

Object Diagram

Composite Structure Diagram

Use Case Diagram

Activity Diagram

State Machine Diagram

Sequence Diagram

Interaction Overview Diagram

Timing Diagram

UML has many types of diagrams, which are divided into two categories:

Some types represent structural information, and the rest represent general types of behavior, including a few that represent different aspects of interactions. These diagrams can be categorized hierarchically as shown below:



**Ans 3** **:-**

**Software engineering** is a field of engineering, for designing and writing programs for computers or other electronic devices. A software engineer, or programmer, writes software and compiles software using methods that make it better quality. Better quality software is easier to use, and the code is easier to understand, to maintain, and to add new features.

**Software engineering** is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

Need of Software Engineering:

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

Large software - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.

Scalability- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.

Cost- As hardware industry has shown its skills and huge manufacturing has lower down he price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.

Dynamic Nature- The always growing and adapting nature of software hugely depends upon the environment in which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.

Quality Management- Better process of software development provides better and quality software product

Characteristics of good software:

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds.

Operational:

This tells us how well software works in operations. It can be measured on:

Budget,Usability,Efficiency,Correctness,Functionality,Dependability,Security,Safety

Transitional:

This aspect is important when the software is moved from one platform to another:

Portability,Interoperability,Reusability,Adaptability.