Day 1:

Topic:

OOPS

Class

Objects

Encapsulation

Abstraction

Polymorphism

Inheritance

Assignment:

- Create a class for vehicle which contains vehicleType (petrol, diesel ,ev ,…), brand , model, color, mileage, price.
- Derive car and bike from vehicle
- Car should have attributes no of persons, carType (suv, sedan, etc) ,
- Bike should have attributes weight, bikeType(scooter, motorbikes ,etc)
- Create a abstract method getNoOfWheels in vehicles which has to be overridden in derive class
- Using operator overloading (<<) to display, brand, model, type, color, price.
- Using operator overloading (< (less than) or > (greater than)) two vehicles can be compared, return true or false based on price of the vehicle.
- In main function , create some objects for car and bike , print the noofWheels , compare two vehicles.

Day 2 :

Topic:

Overloading

Overriding

Assignment:

Bank In this exercise our goal is to use the Overriding and Overloading concept to implement Comparisons of Different Banks.

- We should have two Base/Parent class named Bank, Loan.
- The Bank class should have the methods like getName, getEstablisedDate , BankType(private/public), getInfo(print all details of the bank), branch name.
- The Loan class should have methods like getAvailableLoans (goldLoan, landLoan, personalLoan) , get Interest Rate, documentRequeired for specific loan.
- We should Create Multiple Classes like (HDFC, SBI, ICICI, Indian Bank) derived from Bank and Loan class and Override the methods of the Bank and Loan classes.
- Create a Class named Broker to compare the interestRate of Banks using overloading. Below cases should be handled.
  - Compare Bank Interest rate and Display the bank which has low interest rate
  - Compare two banks loan rates
  - Compare three banks loan rates
  - Compare “n” banks loan rates (Parameter as Array).

In addition to the above comparison this class should also have methods to

1. Print the Details of single bank
2. Print the details of the multiple banks

Day 3 :

Topic:

Pointer

Smart Pointers

Shared Pointer

Unique Pointer

Assignment:

A city has many buildings, which are arranged in a matrix order of size N * M. Some buildings are empty some have a family living in it. we can denote these buildings as a N * M matrix with number of person in a family as values, empty building will be denoted with zero. Each family in the building plans to form a group, a group can be only formed with adjacent neighbours vertically or horizontally.

Given an input matrix of size N * M. Have to find the number of groups that can be formed and the members count in each group. Write a function that will return an array with group members count and number of groups formed.
The group with more members will take leadership, so write
another function to find the leader group.

Sample Input 1:

Matrix : [0, 2, 0, 3, 1,
4, 1, 2, 0, 0,
2, 0, 0, 0, 0,
0, 1, 0, 2, 3
0, 0, 1, 1, 0]

Sample Output: [11, 4, 1, 7]
Leader group : 11

Note:

1. Do not use STL (like, vector, map etc..)
2. Use dynamic memory allocation for arrays.
3. The program should exit without memory leaks.

Explanation:

0, 2, 0, 3, 1,
4, 1, 2, 0, 0,
2, 0, 0, 0, 0,
0, 1, 0, 2, 3
0, 0, 1, 1, 0

In the above matrix, the groups formed are {2, 4, 1, 2, 2}, {3, 1}, {1}, {2, 3, 1, 1}. Summing up the family members we get total members of a group as output.
Thus the output is, [11, 4, 1, 7]
Leader group : 11.

Optional :

The leader group need to take control of all other groups, so it need to find a path to other groups. Have to find the shortest distance between leader and other groups.

Sample Output: [1, 1, 2].

Explanation:

0, 2_____ 3, 1,
4, 1, 2, 0, 0,
2, |, | , 0, 0,
0, 1, | , 2, 3
0, 0, 1, 1, 0

As denoted above, if a leader group is denoted as {11}. Thus, the distances found are
{11} to {4} is 1
{11} to {1} is 1
{11} to {7} is 2

Day 4 :

Topic:

Constructor

Destructor

Assignment:

Design a class named Rectangle with members left, top, width, height with int* datatype.

1. Create a default constructor which initializes all the values with zero.

All Parameters in a parameterized constructor will have non pointer data type (eg. int, float etc..)
2. Create a parameterized constructor with parameters width, height. where left, top will be initialized with zero.
3. Create another parameterized constructor with all values as parameters.
4. Create another parameterized constructor with all values as parameters with float data type. Float data type should be rounded off to nearest int value and set. (1.6 will be rounded to 2 && 1.3 will be rounded to 1).
5. Write a copy constructor that will copy all the values from one obj to other.
6. Above class will have another member variable named area with int* data type. where area will be set once the variable is copied to another. (Have to find the area and set It in copy constructor).
7. Create a destructor, which frees the memory.
8. Write a function to print the rect values.

Program should exit without memory leaks.

Day 5 :

Topic:

Namespace

Assignment:

- ⭐ Create class inside namespace
- ⭐ Create Nested namespace with math equations
- ⭐ Import it in a separate class
- ⭐ Use inline namespace calling
- ⭐ Namespace aliases

Day 6 :

Topic:

Exception handling

Assignment:

1. The exceptions the code may throw along with the handler message are listed below:
Division by zero: Print "Invalid division".

Creation of heap memory variable: If the computer is unable to allocate the memory for computation, Print 'Memory is compromised!'

String parsed to a numeric variable : Print "Format mismatch".

Accessing an invalid index in string : Print "Index is invalid".

Accessing an invalid index in array : Print "Array index is invalid".

MyException (user defined exception) : This is a user defined Exception which you need to create. It takes a parameter. When an exception of this class is encountered, the handler should print "MyException[param]", here is the parameter passed to the exception class.

Exceptions other than mentioned above : Any other exception except the above ones fall in this category. Print "Exception encountered".

Finally, **after the exception is handled, print "Exception Handling Accomplished"**.

NOTE: To obtain array values, string and index (to be searched for in the string) as input and perform division operations ,etc to handle exceptions respectively.

2. Cases to be implement (another prog)

- If an exception occurs within outer try block.
- If an exception occurs inside inner try block.
- In case of no such notable exceptions.
- To re-throw an exception already catched.

3. Use case based question 1

Create a class Student with members roll no, name, age and course. Initialize values through parameterized constructor.If age of student is not in between 15 and 21 then generate user-defined exception "AgeNotWithinRangeException". If name contains numbers or special symbols raise exception "NameNotValidException". The course must fall under a predefined list of available courses (not case sensitive), if not "InvalidCouseException" must be raised. Define the 3 exception classes. The user can only create one object. If user attempts to create more than one object for Student class, they should not be able to create that is we must throw an exception in that case.

Day 7 :

Topic:

Lambda Functions

Assignment:

There are multiple devices in a house, which are connected to the device of the user. The user has a client application running on his device that needs to know the sensor status of the connected devices.

Implement a cli application that captures all these callbacks and reports the status of the devices in "real time". Show the status of the individual sensors values in a separate column.

```
SENSORS     STATUS
-------     ------
Fan         OFF
Light       ON
Temp        22.0
Door        OPEN

1. Simulate Input
2. Device Automation
```

Client application should handle the following events for all devices and sensors,
`onConnect()`
`onDisconnect()`

Every device should also provide additional events based on its capability. For example, a temperature sensor should provide `onTemperatureChange(float newTemperature) event` and a device such as DoorControl should provide the `openDoor()` and `closeDoor()` controls.

DEVICE AUTOMATION

The client application should be capable of automating certain tasks given by the user. The following syntax should be followed when getting input from the user.

```
// syntax
If: <sensor_name> <comparison> <sensor_value>
Then: <device> <function>

// For example:
If: Temperature > 22.0c
Then: Fan turnOn
```

When the user enters the automation, store them in vectors of std::function for >,<,= conditions and fire them when activated by keypress and the conditions given in the automation menu is met.

SIMULATING INPUT

These key codes can be followed for simulating the input sensors

1. Increase Value
2. Decrease Value
3. Connect Device
4. Disconnect Device

At least the following sensors/devices should be handled,

- Temperature sensor
- Motion sensor
- Water level sensor
- Gas detection sensor
- Fan
- Light
- Door

Day 8 :

Topic:

Templates

Assignment:

Consider Zoho provides telecommunication services like landline and mobile. You are to program a common billing system using generic templates for these services.

1. Create a class for mobile connection. Each mobile connection should have a ten digit mobile number and a bill amount to be paid.
2. Create a class for landline connection. Each landline connection should have a six digit landline number, STD code and a bill amount to be paid.
3. Create a class with template to process bill payments. The class should do the following:
    1. Store all connections (for any particular service).
    2. A function to pay bills using the connection number. The function should update the bill due amount to zero for the particular connection. This function should call a function with same name in mobile and landline to find the connection using the number (to demonstrate templates). For mobile, match the user input with the ten digit mobile number. For landline, user will input STD code and the six digit number together. You will have to separate them and find the corresponding connection.
    3. A function to update new due bill amount using the connection number. If there is already amount due for a particular connection, the new amount should get added. This function should also call the common functions

You have to provide the user with the following options:

1. Add a new mobile connection.
2. Add a new landline connection.
3. Pay bill using the connection number.
4. Update bill amount using the connection number.