Date : ..... Page No. 1

Assignment No. : 01

Assignment Topic : Basic Concepts of oops

1. Explain about the Basic Concept of oops?

A. Introduction to object oriented programming?:-

1. Object oriented programming refers to languages that use objects in program.
2. Object oriented programming aims to implement real world entities like Inheritence, hiding, polymorphism etc.
3. The main aim of object oriented programming is to bind together the data and the functions that operate on them.
4. object oriented programming is a programming paradigm based on the concept of objects which can contain data and code.
5. c++ and java follows object oriented programming approach.

Concepts of oops :-

1. Object :-

An Instance of a class is called object or an object is any-thing in the real world.

An object is Basic run time entities in an object oriented programming. In object consists 3 types of features such as.

State :-

It is represented by properties (or)

Attributes (or) Instance Variables.

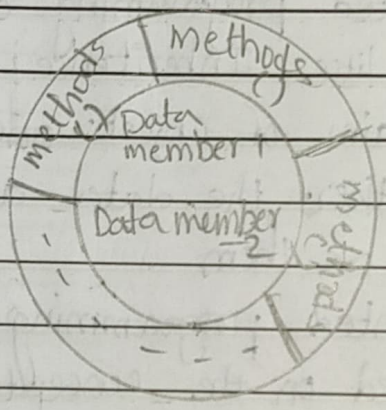
Behaviour:-

It is represented by Actions (or) method (or) functions.

Identity:-

It is unique name given to the created object.

Syntax:-



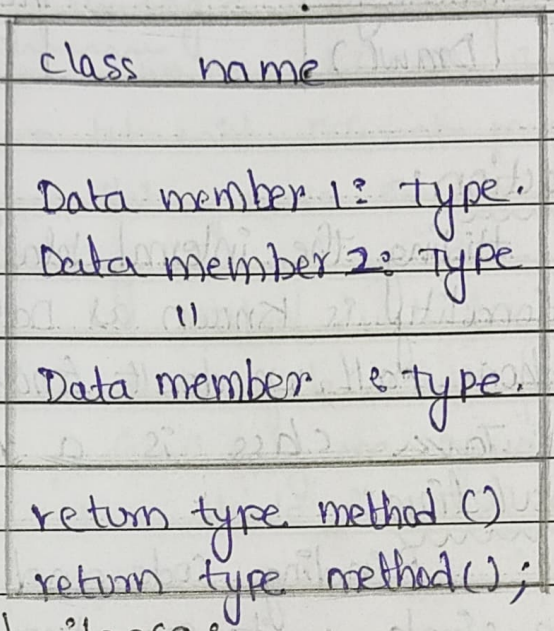
→ Architecture of object.

Object

2. class:-

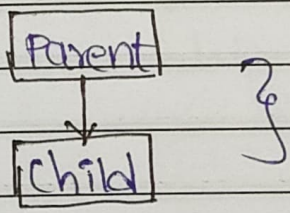
class is a user defined blue-print or prototype which objects are created. Actually class is a logical view of entity. Once class has been defined, we can create no. of class related objects. In that class consists 3 parts such as 1. Class name, 2- Data member, 3. methods

Class Architecture :-



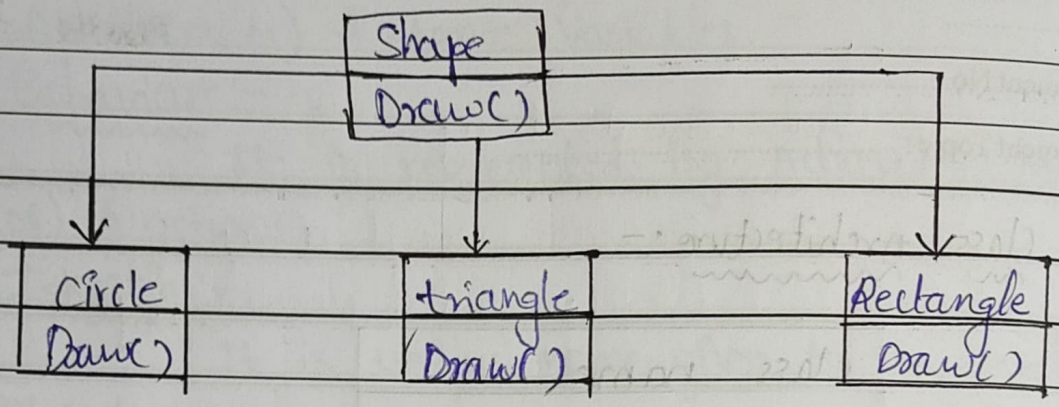
3. Inheritence :-

The new class acquires all the properties and behaviours of an existing class. It is known as inheritence. It provide code reuseability.



4) Polymorphism :-

If one task is perform in different ways it is known as polymorphism.



5. Data Abstraction:-

Hiding the internal details and showing functionality is known as Data Abstraction. For example phone call, we don't know the internal processing. In Java class is a ADT.

6. Data Encapsulation:-

Binding code and Data together into a single unit are known as Encapsulation.

Dynamic Binding:-

when type of object determined at Runtime then it is called dynamic binding.

Date : ..... Page No. 5

Assignment No. : 2

Assignment Topic : over Loading

2. Explain about method overloading in java?

A. method overloading :-

We can define multiple methods with the same name but with different parameters of parameters type. then this process is known as method overloading.

Method overloading is a type of static polymorphism or compile time polymorphism. If we have to perform only one operation, having same name of the methods increases the readability of the program.

Advantages :-

Method overloading increases the readability of program.

Different ways to method overloading :-

* By changing the no. of parameters.
* By changing the datatypes.

Program :-

```
class Rectangle
{
float l,b;
void rect (float l, float b)
{
  this.l=l;
  this.b=b;
float ar = l*b;
```

```
float p = 2*(l+b);
System.out.println("Area of rectangle =" +ar);
System.out.println("parameter =" +p);
}
    void rect (int l, int b).
{
    this.l=l;
    this.b=b;
float this ar = l*b;
float this p = 2*(l+b);
System.out.println("Area of rectangle =" +ar);
System.out.println("perimeter =" +p);
}
void rect (float l)
{
    this.l=l;
        b=l;
System.out.println("Area of rectangle =" +(l*b));
System.out.println("perimeter =" +(2*(l+b)));
}
public static void main (String args[])
{
  Rectangle v = new Rectangle ();
   v.rect (4.2F);
   v.rect (4,5);
   v.rect (4.2F, 2.5F);
}
}.
```

Date : ..... Page No. 7

Assignment No. : 3

Assignment Topic : Inheritance.

3. What is various types of inheritance in java?

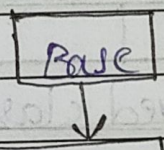
A. Inheritance :-

the processing of acquiring the qualities one class to another class is called inheritance. this qualities include both properties and functions of that class. using inheritance, the "extends" ~~function~~ the features of a class into another class. In java, extend is a keyword. There are different types of inheritance.

1. Single inheritance
2. Multilevel inheritance
3. Hierarichal inheritance.
4. multiple level inheritance

1. Single inheritance :-

Deriving a new class based on only base class and the newly derived class is not other deriving new class is called single inheritance.



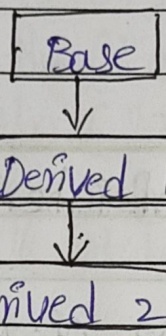
Super / Existing / parent class.

-subclass / new / child class.

2. multilevel inheritance :-

Deriving a class based on base class which is already derived from another class is called multilevel inheritance.

Base
Derived 1
Derived 2



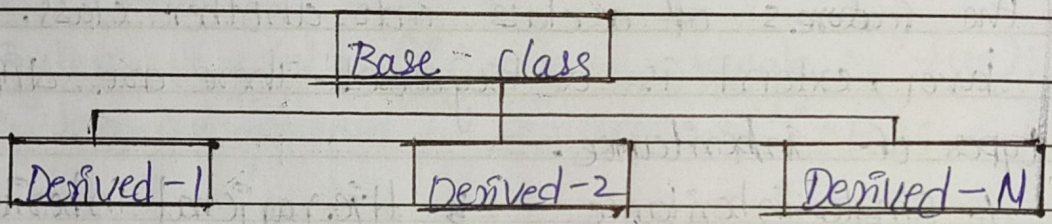
Super / Existing / parent class

subclass / new / child / parent / super

sub / new / child class

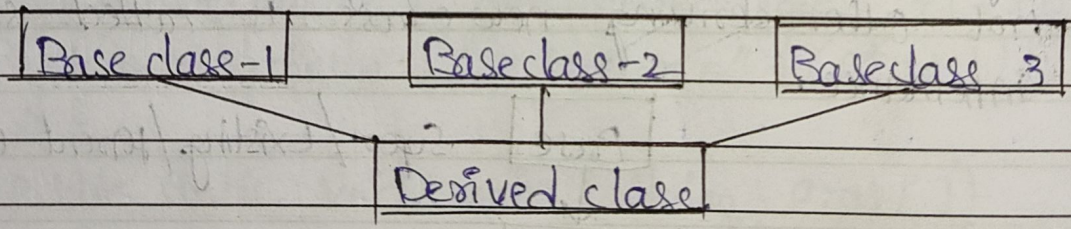
Hierarchical Inheritance :-

Deriving a class based on existing class which is derived from "base class".



Multiple inheritance :-

Deriving a new class based on multiple base class is called multiple inheritance.



In java it is not possible to deriving a new classes based on multiple base classes rather than we use interface to achieve these techniques.

Date : .....

Page No. 9

Assignment No. : 04

Assignment Topic : Exception handling.

4. Explain Exception Handling in Java?

A) Exception handlers :-

An exception is a condition caused by runtime error in a program. whenever the java interpreter finds the runtime error it creates an exception object. These Exception objects handles Exception handlers.

Then exception handling code basically consist of 2 segments, one to detect errors and throw exception and second is to catch the exception and take appropriate actions.

Types of exception Handlers :-

They are 3 types of exception handlers :-

1. Try block.
2. catch block
3. finally blocks.

1. Try blocks :-

The try is a keyword in java it is used to perform detect an error and throws an exception to the catch blocks.

Syntax :-

```
try
{
    caused error statement
}
```

2. catch block :-

The catch is a keyword in java. It is used to perform catch the error object (exception) & handles that exception thrown by the try block. If there are more than 1 catch blocks are defined for a single try blocks, then appropriate catch block will receive the exception thrown by the try block.

Syntax:-

```
Catch (Error type Exception Error object 1)
{
  Exception Handling
}
- - - -
- - - -
catch (error type Exception Error object n)
{
  Exception handling
}
```

3. Finally block :-

The finally is a keyword. It is used to perform reallocate the program. It specify a block of statement to execute when none of the catch blocks call the exception. It is optional in exception handlers. A try block can have only one finally block.

Syntax:-

```
finally
{
  Re-allocate statements
}
```

Date : .....

Page No. 11

Assignment No. : 5

Assignment Topic : Interface.

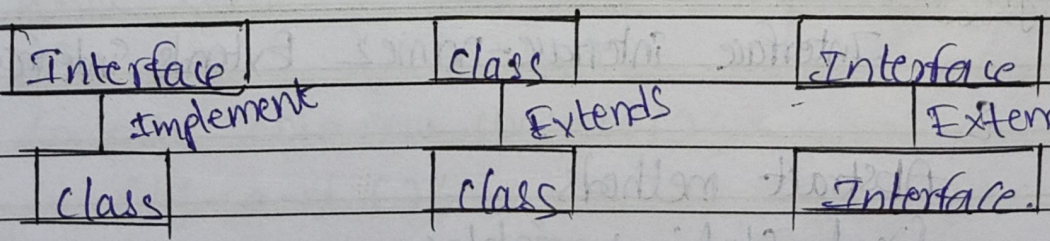
5. write about Interface concept in java?

A) Interface :-

An interface in java is a blue print of a class. An interface is a mechanism to achieve data abstraction. It has final static variables and Abstract method. There can be only abstract methods in an interface, not method body. This method body write in classes. It is used to achieve data abstraction and multiple inheritance in java.

Rules for interface :-

1. All the methods of an interface are public and abstract.
2. All the variables of an interface are final, static and public.
3. Interface method should not be static and final.
4. An interface can extend one or more other interface.
5. An interface can't implement other interface.
6. An interface can implement no. of classes.



Syntax:-

```
interface interface_name
{
[final static members]
[Abstract members]
}
```

Implementing an interface:-

It is a collection of abstract methods. We can't give a life.

It must be implemented in a class. Through classes, we can give life to the interfaces. Here, uses "implement keyword" in class.

Syntax:-

```
class classname implements interface-name
{
class definition
}
```

Extending an interface:-

like classes, interface can also be extended. An interface can extend the features one interface to another interface. Here features include abstract methods and final static variables.

Syntax:-

```
Interface interface_name2 Extends interface name1
{
Abstract methods
final static variables
}
```

Date : .....

Page No. 13

Assignment No. : .....

Assignment Topic : .....

Program :-

```
Interface myinterface 1
 {
  final static int a=4, b=2;
 }
Interface myinterface 2 extends myinterface1
 {
   Public void add ();
   Public void mul ();
 }
class compute implements myinterface 2
{
 Public void add ()
 {
  System.out.println ("the sum of two values = "+(a+b));
 }
Public void mul ();
 {
System.out.println ("the mul of two values = "+(a*b));
 }
Public static void main (String []args)
 {
 Compute obj = new compute ();
    Obj.add ();
    Obj.mul ();
 } }.
```

Date : .....

Page No. 15

Assignment No. : 6

Assignment Topic : Packages

6. Explain about packages in java?

A. A java package is a group of similar type of classes, interfaces, sub-packages. A java package is a sub-directory. package in java two categories. package is a keyword in java.

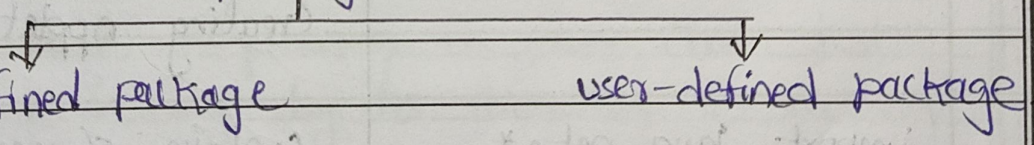
1. Built in package.
2. user defined package.

Advantages of package :-

1. Java package is used to categories the classes and interfaces.
2. It can be easily managed
3. java package provides access protection.
4. Java package removes "name-collision".

Types of packages :-

There are two types of packages



Predefined package :-

pre-defined package consists large no of classes which are part of java API.

Some of the commonly used predefined packages.

Package name	Usage.
Import java.util.*	contains utility classes such as scanner, Date & time and linked list operation.
import java.io.*	contains classes for input & output operations.
import java.lang.*	contains language support classes such as primitive data types, mathametical operations.
Import java.awt.*	contains classes for implimenting the components such as buttons, check boxes, menus and so on...
import.java.Applet.*	contains classes for creating Applet
import java.net.*	contain classes for supporting networking operation.

2. user-defined package :-

These are designed or created by the developer to categories classes and packages. They are much similar to the built in packages. It can be imported into the other classes and used the same as we use predefined packages.

To create a package, we are used to the package keyword with package name

Syntax :-

```
package <package_name>;
```

Ex :-

```
package mypackage;
```

steps to creating user-defined package :-

Step1 :

Creating a package in java class. just write a package by its following its package name.

Step2 :

Both class name and java files name must be same.

Step 3:-

After compilation, it generates the class file with package and then import the package to the another class. Here "import" is a keyword.

Date: .....

Page No. 19

Assignment No.: 07

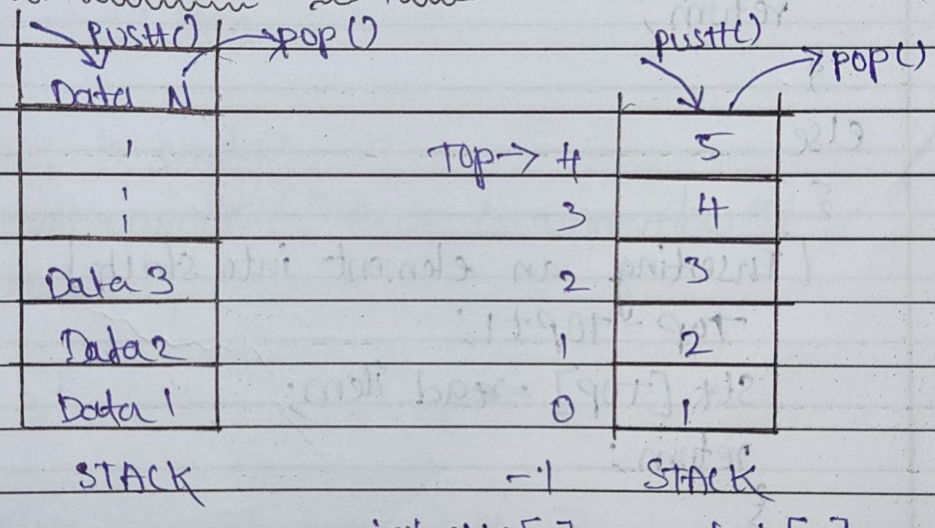
Assignment Topic: Stack Algorithm.

7. Explain about stack Algorithm?

A. Stack :-

A stack is a linear order collection of elements in which elements are inserted and deleted at only one end point. Inserting an element at top of the stack is called "push()" and deleting an element at top of the stack is called "pop()". It has only one pointer called "top", which means indicate the indexes of the stack. An order or principle of stack is "LIFO".

memory representation of stack :-



int stk[] = new int[5]

Implimentation of stack on PUSH() operation :-

PUSH() :-

Push operation is used to perform inserting an element into the top of the stack.

overflow():-

If stack is full the flash the message is "STACK IS OVERFLOW".

TOP :-

top is a pointer and it indicates the indexes of top of the element. In PUSH() operation TOP is increased.

Algorithm of PUSH():-

```
Algorithm PUSH (Stk[max], max=5, TOP=-1, item)
If (top==max-1) then
{
 [STACK is FULL]
system.out.println ("Stack is overflow");
 return;
}
else
{
 [Inserting an element into stack]
    TOP=TOP+1;
    Stk.[TOP] =read item;
    return;
   }
}
```

Implimentation of stack on pop operation:-

POP():-

pop operation is used to perform deleting an element from the top of the stack.

underflow() :-

If doesn't have any element in the stack then flash the message is "STACK IS UNDERFLOW".

TOP () :-

It is a pointer and it indicates the indexes of top of the element. In pop() operation, TOP is decreased.

Algorithm of pop() :-

```
Algorithm pop (stack[max], max=5, TOP=max-1)
{
if (top==-1) then
  {
   [stack is empty]
System.out.println("stack is underflow")
  return.
}
else
  {
   [Deleting an element in stack]
System.out.println("Deleting an item is => "stk[top])
   top=top-1
    return
  }
}.
```

Date : .....

Page No. 23

Assignment No. : 08

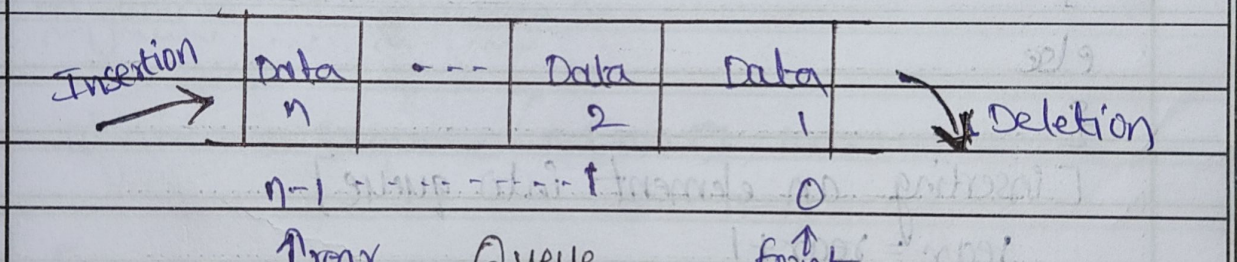
Assignment Topic : Queue Operations.

8. Explain about queue operation?

A. Queue :-

Queue is a linear order collection of elements in which element are inserted at one end and deleted at other end. An inserting an element at the rear of the queue is called "Insertion operation" and deleting an element at the front of the queue is called "deletion operation" it has two end pointers called "Front and rear". An order or principle of the queue is "LIFO" (last in first out).

Memory Representation of queue :-



Implementation of queue on Enqueue operation :-

Enqueue [ ] :-

Enqueue operation is used to perform inserting an element into the rear of the queue.

Is full :-

If queue is full then can't insert any element into the queue. Here, flash the message is "Queue is full".

Rear :-

Rear is the "pointer" and it indicates the indexes in enqueue operation rear pointer is increased by "1".

Algorithm of Queue :-

```
Algorithm Enqueue (q[max], max=4, front=-1, item)
{
  if (rear == max-1)
  {
    [Queue is full)
  System.out.println("queue is full -- can't insert any
                      elemt");
    return
  }
  else
  {
  [inserting an element into queue].
    rear = rear+1
    q(rear) = read item
    if (front == -1)
    {
      front = 0
    }
    return
  }
}
```

Date : .....

Page No. 25

Assignment No. : .....

Assignment Topic : .....

Implementation of queue on Dequeue operation :-

Dequeue () :-

Dequeue operation is used to perform deleting an element at front of the queue.

Is empty :-

If queue is empty then we can't delete here. Display the message is "empty".

Front :-

Front is a pointer and it indicates the starting index(0) in dequeue operation front increased by 1.

Algorithm of dequeue :-

```
Algorithm Dequeue (Q[max], max=n, front=0,
                   rear=n-1)
{
 if (front == -1 && rear == -1)
   {
    [Queue is empty]
System.out.println ("Queue is empty .. can't Delete")
    return
   }
else
 {
  [Delete an element from the queue]
```

```
System.out.println("the deleted item => Q[front]")
  front = front + 1
   if (rear + 1 == front)
   {
      front = -1
      rear = -1
   }
return
  }
}
```

Date : .....

Page No. 27

Assignment No. : 09

Assignment Topic : Binary Tree Traversals

9. Explain about the Binary Tree Traversals?

A. Binary Tree traversals :-

Binary tree is a finite set of elements called nodes, in which each node consists of two child notes (left, right) or zero (0) or (1). The Binary tree traversals is classified into 3 ways.

1. Pre-Order
2. In-Order
3. post-order.

1. pre-order :-

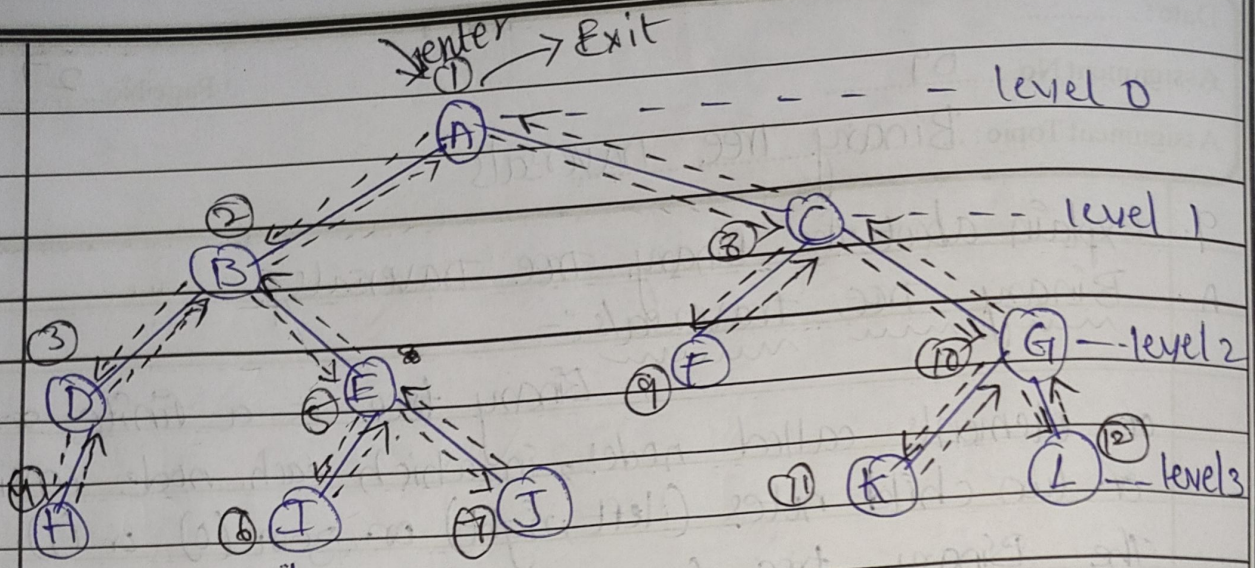
In a pre order traversal, each root node is visited before its left and right subtrees are traversed the pre order search is also called as "Back tracking". the steps for traversing a binary tree in pre-order traversal.

steps for pre-order :-

step 1 : visit the root node (display the data)
step 2 : traverse the left subtree
step 3 : Traverse the right sub tree.

Order :-

ROOT → LEFT SUBTREE → RIGHT SUBTREE



Procedure :

A → B → D → H → E → I → J → C → F → G → K → L

② In-order :-

In a In-order traversal, the root node of each subtree is visited after its left subtree has been traversed but before the traversal of its right subtree begins. the steps for In-order traversals.

steps for In-order :-

Step 1 : Traverse the left subtree
Step 2 : visit the root node (Display the data)
Step 3 : traverse the right subtree

order :-

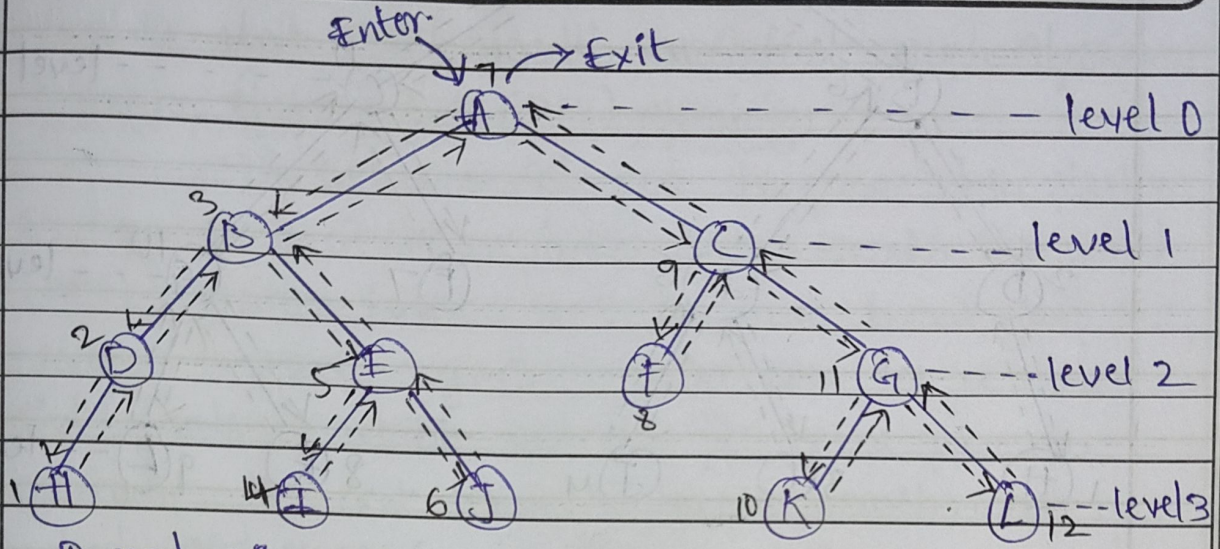
LEFT SUBTREE → ROOT → RIGHT SUBTREE

Date : .....

Page No. 29

Assignment No. : .....

Assignment Topic : .....



Procedure :-

H → D → B → I → E → J → A → F → C → K → G → L

③ Post-order :-

In a post-order traversal tree. First traversal the left subtree and travers. the right subtree after that visit the root node. the steps for post-order traversal is as follows.

Steps :-

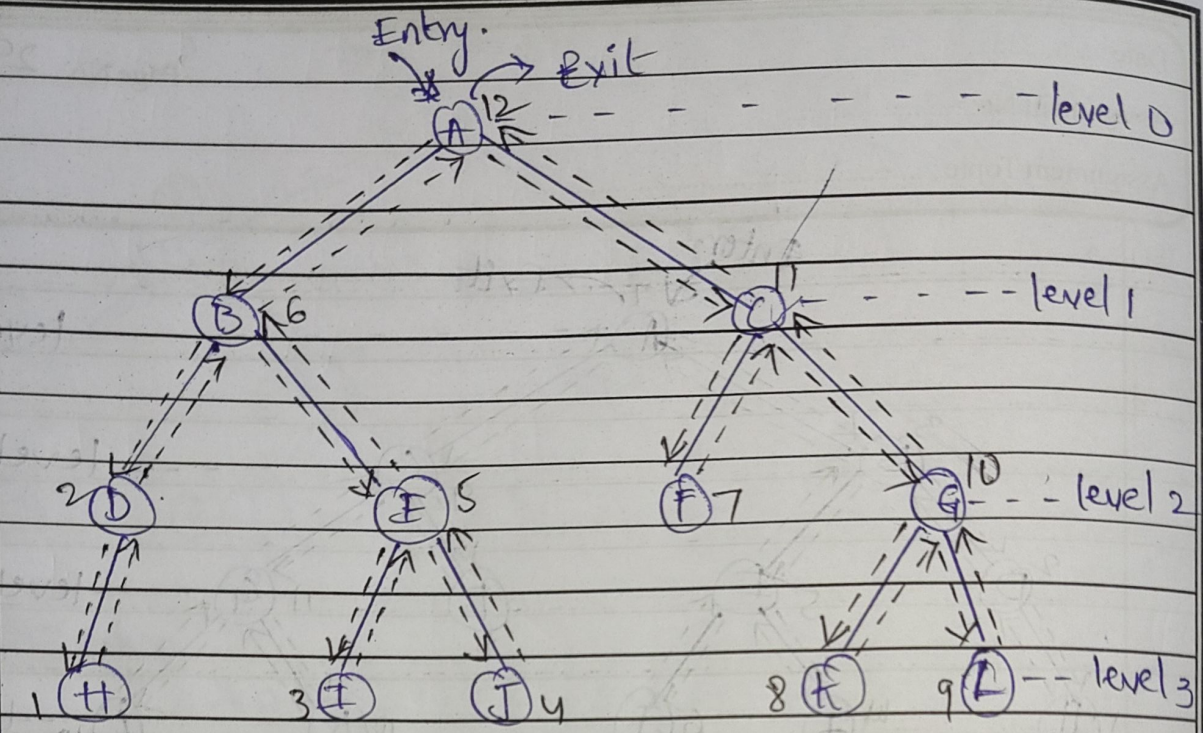
Step1 : Traverse the left subtree

Step2 : Traverse the right subtree

Step3 : Visit the root node.

Order :-

LEFT SUBTREE → RIGHT SUBTREE → ROOT



Post-Order:

H → D → I → J → E → B → F → K → L → G → C → A.

Date: .....

Page No. 31

Assignment No.: 10

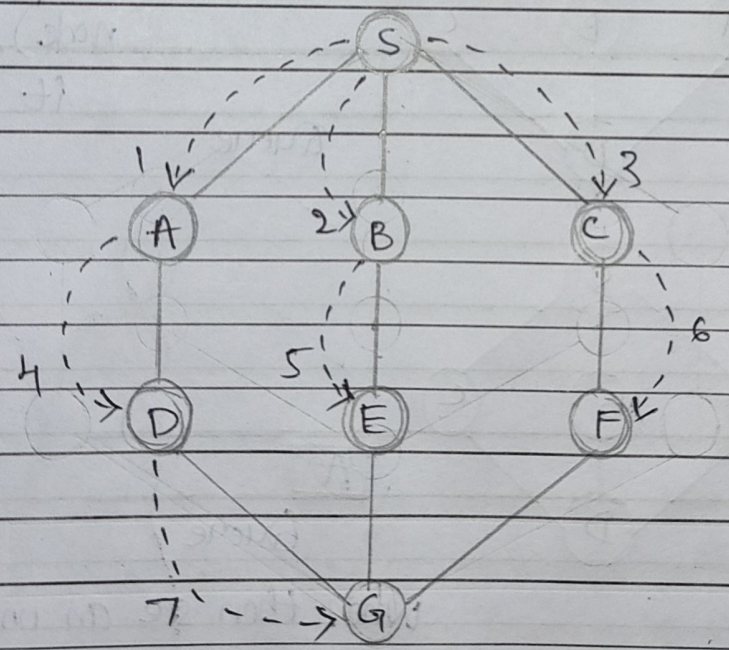
Assignment Topic: BFS Algorithm.

10- Explain about (BFS) Breadth search first search Algorithm?

A: Breadth first search (BFS) Algorithm:-

The DFS traversal uses the queue data structure to keep track of the unvisited nodes. Breadth first search (BFS) algorithm traverses a graph in a breadthward motion to search a graph data structure for a node that meets a set of criteria. It uses a queue to remember the next vertex to start a search, when a dead end occurs in any iteration.

Breadth first search (BFS) algorithm starts at the tree root and explores all nodes at the present depth prior to moving on to the nodes at the next depth level.

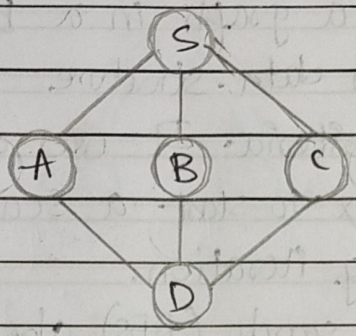
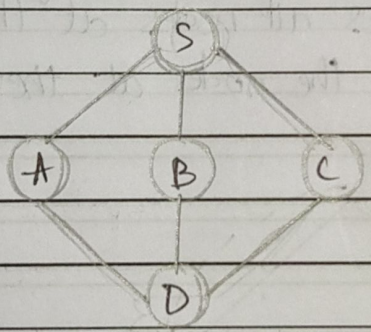
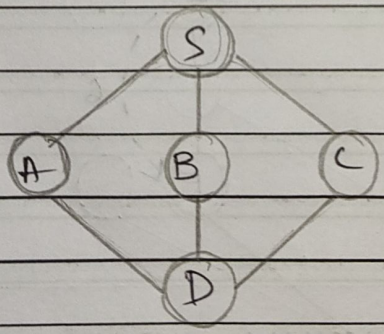


It employs the following rules.

- Rule 1- visit the adjacent unvisited vertex. mark it as

visited. Display it. Insert it in a queue.

- Rule 2 – If no adjacent vertex found, remove the first vertex from the queue.
- Rule 3 – Repeat Rule 1 and Rule 2 until the queue is empty.

Step	Traversal	Description
1.	 Queue	initialize the queue.
2.	 Queue	we start from visiting S (starting node), and mark it as visited.
3.	 A Queue	When then see an unvisited adjacent node. from S. In this example, we have three nodes but alphabetically we choose A, mark it as visited and enqueue it.

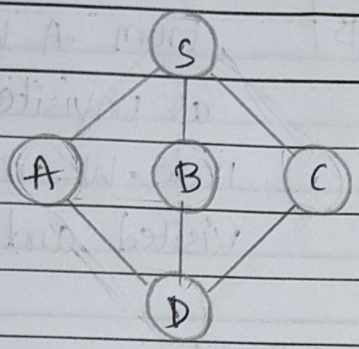
Date : .....

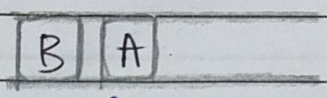
Page No. 33

Assignment No. : .....

Assignment Topic : .....

4.

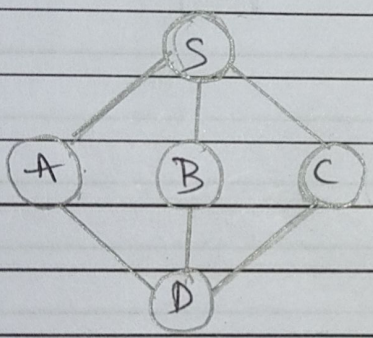


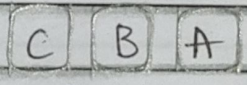


Queue

Next, the unvisited adjacent node from S is B. we mark it as visited and enqueue it.

5.

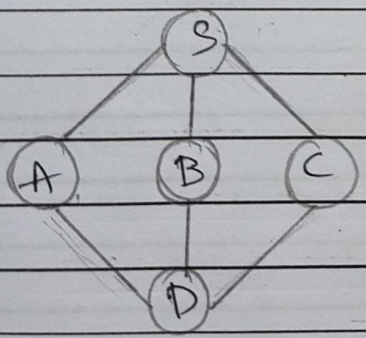


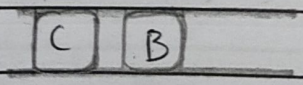


Queue

Next, the unvisited adjacent node from S is C. we mark it as visited and enqueue it.

6.

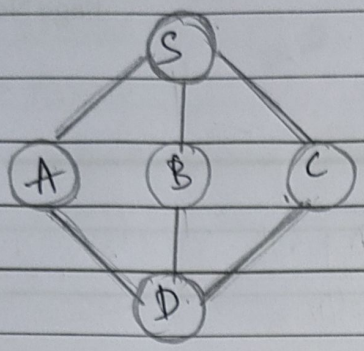




Queue

NOW, S. is left with no unvisited adjacent nodes. so we dequeue and find A.

7.



D	C	B

queue

From A we have D as unvisited adjacent node. We mark it as visited and enqueue it.

At this stage, we are left with no unmarked (unvisited) nodes. But as per the algorithm we keep on dequeuing in order to get all unvisited nodes. When the queue gets emptied, the program is over.