SRINIVAS UNIVERSITY

INSTITUTE OF COMPUTER SCIENCE & INFORMATION SCIENCE

CITY CAMPUS, PANDESHWAR,

MANGALORE - 575 001

Question Bank Answers with Bloom's Model

Data Structures Using C++

B.C.A - III SEMESTER

Compiled by
Faculty

DATA STRUCTURES USING C++ QUESTION BANK

Exam	**III Semester**	**Paper code**	**21CAC – 7**
Subject	**Data Structures Using C++**	**Class**	**II BCA**
Maximum marks	**50**	**Time**	**2 Hours**

Weightage Table

Sl. No	Objectives	Marks	Percentage marks
1.	Knowledge (Remembering)	05	10
2.	Understanding	20	40
3.	Application	15	30
4.	Skill	10	20
Total		**50**	**100**

Blueprint with Unit wise Marks

Unit	Remembering			Understand			Application			Skill			Total
	OT	SA	Unit wise Marks	OT	SA	Unit wise Marks	OT	SA	Unit wise Marks	OT	SA	Unit wise Marks	
1	1(1)	1(4)	**5**	1(1)	1(4)	**5**	-	-	-	-	-	-	10
2	-	-	-	1(1)	1(4)	**5**	-	-	-	1(1)	1(4)	**5**	10
3	-	-	-	1(1)	1(4)	**5**	1(1)	1(4)	**5**	-	-	-	10
4	-	-	-	1(1)	1(4)	**5**	1(1)	1(4)	**5**	-	-	-	10
5	-	-	-	-	-	-	1(1)	1(4)	**5**	1(1)	1(4)	**5**	10
	05			**20**			**15**			**10**			**50**

Blueprint

Unit	Remembering		Understand		Application		Skill		Total
	OT	SA	OT	SA	OT	SA	OT	SA	
1	1(1)	1(4)	1(1)	1(4)	-	-	-	-	10
2	-	-	1(1)	1(4)	-	-	1(1)	1(4)	10
3	-	-	1(1)	1(4)	1(1)	1(4)	-	-	10
4	-	-	1(1)	1(4)	1(1)	1(4)	-	-	10
5	-	-	-	-	1(1)	1(4)	1(1)	1(4)	10
	05		**20**		**15**		**10**		**50**

MODULE 1

Multiple Choice Questions

(Questions for Remembering)

1. ______________is a step-by-step representation of a solution to a given problem.
A. Algorithm
B. flowchart
C. Pseudocode
D. None of the above

2. The Sequence of the Instruction written to perform a specific task is called the_________.
A. Statement
B. Program
C. Algorithm
D. None of the above

3. ____________ is a procedure or step-by-step process for solving a problem.
A. Algorithm
B. Flowchart
C. Pseudocode
D. All of these

4.Assuming int is of 4 bytes, what is the size of int arr[15];?
A.15
B.19
C.11
D.60

5.How many kinds of elements an array can have?
A.Char and int type
B.Only char type
C.Only int type
D.All of them have the same type

6.Elements of an array are numbered as 0,1,2,3 known as ___________.
A. Index values
B. Subscripts of array
C. Members of an array
D. Both A and B

7. Absolute value of integers |-10| and |-12| are ________________
A. 10, 20, -10, -20
B. 0,-10
C. 10,12
D. 0, 12

8. Floor (2.4) + Ceil (2.9) is equal to ____________________
A.4
B.6
C.5
D.none of the mentioned

9. Factorial of a positive integer n is______________________
A. n!=n(n-2)!
B. n!= n(n+2) !
C. n! = (n-2)!
D. n! = n(n-1)!

10. This Loop tests the condition after having executed the Statements within the Loop.
A. while
B. **do-while**
C. for Loop
D. if-else-if

11. To implement sparse matrix dynamically, the following data structure is used ___________
A. Trees
B. Graphs
C. Priority Queues
D. Linked List

(Questions for Understanding)

12. Choose the right C Statement.
A. Loops or Repetition block executes a group of statements repeatedly
B. Loop is usually executed as long as a condition is met
C. Loops usually take advantage of Loop Counter
D. All the above

13. The optimal data structure used to solve the Towers of Hanoi is ______
A. Tree
B. Heap
C. Priority queue
D. Stack

14. What is the number of moves required to solve Tower of Hanoi problem for k disks?
A. 2k – 1
B. 2k + 1
C. $2^k + 1$
D. $2^k - 1$

15. Which of the following data structure can't store the non-homogeneous data elements?
A. Arrays

B.Records
C.Pointers
D.None

16. The logical or mathematical model of a particular organization of data is called as______________
A. data structure
B. data arrangement
C. data configuration
D. data information

17. Recursion uses more memory compared to iteration.
A. True
B. False
C. Both
D. None of the above

18. Which of the following is a linear data structure?
A. Array
B. AVL Trees
C. Binary Trees
D. Graphs

19. Referring an element outside array bounds is a___________________
A. Syntax error
B. Logical error
C. execution time error
D. both b and c

20. Recursion is similar to which of the following?
A. Switch Case
B. Loop
C. if-else
D. if-else-if

21. Which Data Structure is mainly used for implementing the recursive algorithm?
A. Queue
B. Stack
C. Linked List
D. Tree

22. What's happen if base condition is not defined in recursion?
A. Stack underflow
B. Stack Overflow
C. None of these
D. Both a and b

23. Which searching can be performed recursively?
A. Linear search
B. Binary search
C. Both
D. None

24. Which of the following problems can't be solved using recursion?
A. Factorial of a number
B. Nth fibonacci number
C. Length of a string
D. Problems without base case

25. Recursion is similar to which of the following?
A. Switch Case
B. **Loop**
C. if-else
D. if-else-if

Questions carrying 4 marks

(Questions for Remembering)

1. **What are the data structure operations?**

The following are the 4 data structure operations.

- **Traversing -** accessing each record exactly once so that certain items in the record may be processed.
- **Searching** - finding the location of the record with a given key value orfinding the locations of all records which satisfy one or more conditions.
- **Insertions** - adding a new record to a structure.
- **Deleting -** removing a record from a structure.
- **Sorting** - arranging the records in some logical order.
- **Merging -** combining the records in two different sorted files into a single sorted file.

2. **Describe a note on ADT.**

With an ADT, we know what a specify data type can do, but how it actually does it is hidden. ADT consists of a set of definitions that allow us to use the functions while hiding the implementation. Weencapsulate the data and the operations on data and then we hide them from the user.

3. **Describe sub algorithms.**

A sub algorithm is a complete and independently defined algorithmic module whichis used by some main algorithm or by some other sub algorithm. A sub algorithm receives values called arguments from a calling algorithm; performs computations and then sends back the result to the calling algorithm.

The main difference between the format of a subprogram and that of an algorithm is
Here NAME refers to the name of the algorithm which is used when the sub transmit data
between the sub algorithm and the calling algorithm.

Another difference is that sub algorithm will have a return statement rather than an Exit statement. This means that control is transferred back to the calling program when the execution of the subprogram is completed.

Sub algorithms fall into 2 categories: function sub algorithms and procedure sub algorithms. The major difference between the two is that the function sub algorithm returns only a single value to the calling algorithm whereas the procedure sub algorithm may send back more than one value.

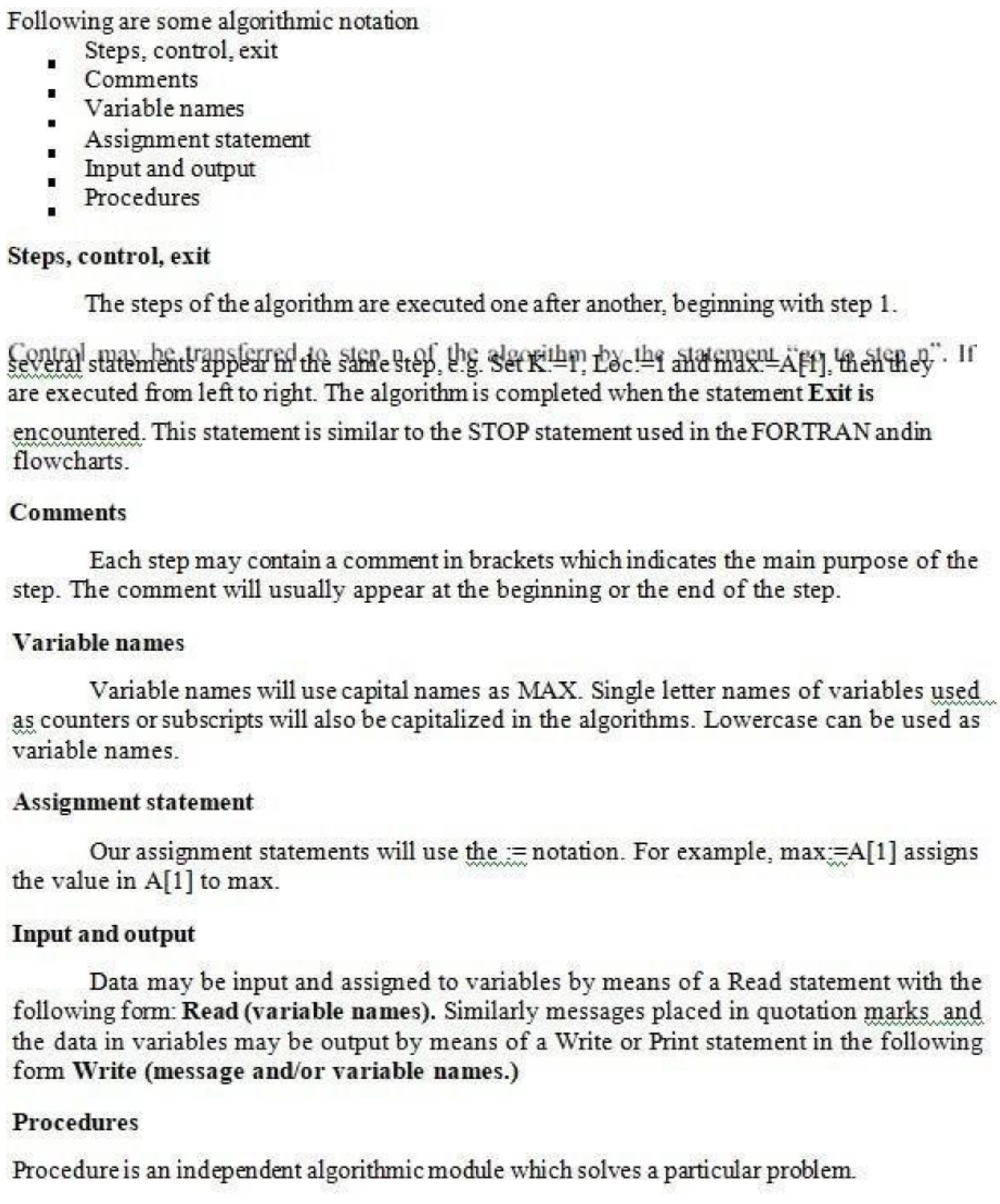
The following function sub algorithm MEAN finds the average AVE of three numbersA, B and C.

```
Mean (A,B,C)
1.Set AVE:=(A+B+C)/3
2.Return (AVE)
```

Note that MEAN is the name of the sub algorithm and A, B and C are the parameters. The return statement includes the variable AVE whose value is returned to the callingprogram.

4. Describe algorithmic notations.

An algorithm is a finite step-by-step list of well-defined instructions for solving a particular problem. For example,
A nonempty array A with N numerical values is given. This algorithm finds the location LOC and the value MAX of the largest element of A. the variable K is used as a counter.



Following are some algorithmic notation

- Steps, control, exit
- Comments
- Variable names
- Assignment statement
- Input and output
- Procedures

Steps, control, exit

The steps of the algorithm are executed one after another, beginning with step 1.

Control may be transferred to step n of the algorithm by the statement "go to step n". If several statements appear in the same step, e.g. Set K:=1, Loc:=1 and max:=A[1], then they are executed from left to right. The algorithm is completed when the statement **Exit is** encountered. This statement is similar to the STOP statement used in the FORTRAN andin flowcharts.

Comments

Each step may contain a comment in brackets which indicates the main purpose of the step. The comment will usually appear at the beginning or the end of the step.

Variable names

Variable names will use capital names as MAX. Single letter names of variables used as counters or subscripts will also be capitalized in the algorithms. Lowercase can be used as variable names.

Assignment statement

Our assignment statements will use the := notation. For example, max:=A[1] assigns the value in A[1] to max.

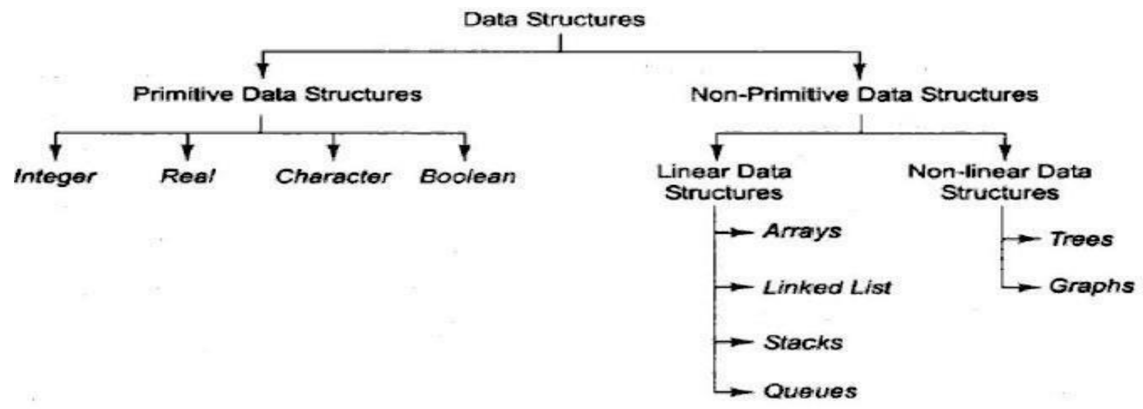
Input and output

Data may be input and assigned to variables by means of a Read statement with the following form: **Read (variable names).** Similarly messages placed in quotation marks and the data in variables may be output by means of a Write or Print statement in the following form **Write (message and/or variable names.)**

Procedures

Procedure is an independent algorithmic module which solves a particular problem.

5. Explain the various types of Data Structures with examples.



Data structures are generally classified into primitive and non-primitive data structures.

Primitive data structures:
Primitive data structures are the data structure that can be manipulated directly by the machine instructions. For example, Basic data types such as integer, real, character and Boolean are known as primitive data structures. These data types consist of characters that cannot be divided and hence they are also called simple datatypes.
Examples: Integers, Characters, Boolean, Real

Non-primitive data structures:
Non-primitive data structures are the data structure that cannot be manipulated directly by machine instructions. For example, the simplest example of non-primitive data structure is the processing of complex numbers.
Linked lists, Stacks, Queues, Trees and Graphs are examples of non-primitive data structures.

6. Explain the classification of Non-primitive data structures with examples.

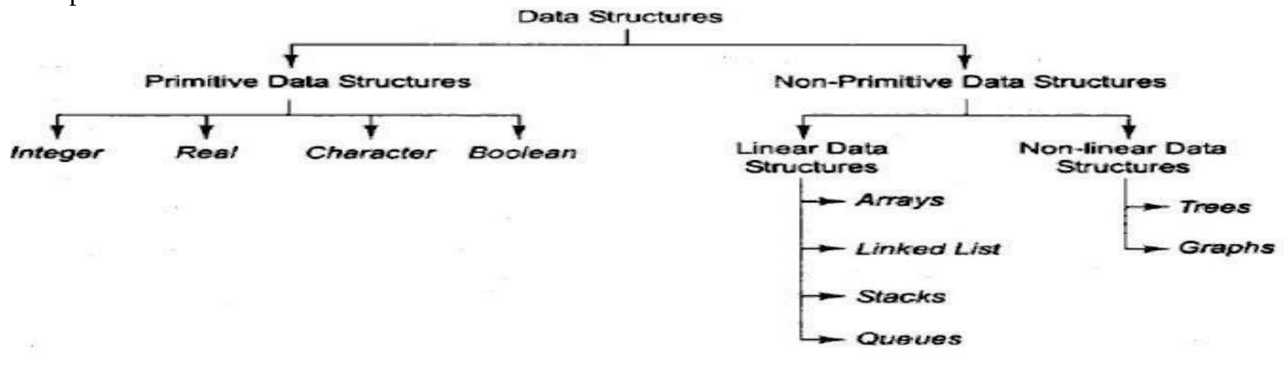
Non-primitive data structures:
Non-primitive data structures are the data structure that cannot be manipulated directly by machine instructions. For example, the simplest example of non-primitive data structure is the processing of complex numbers. Linked lists, stacks, queues, trees and graphs are examples of non-primitive datastructures.

- **Linear Data Structure**

A data structure is said to be linear if its elements form a sequence or a linear list. .Arrays, linked lists, stacks and queues are examples of linear data structures

- **Non-Linear Data Structure**

Similarly, a data structure is said to be nonlinear if the data is not arranged in sequence.Tress and graphs are examples.



(Questions for Understanding)

1.Describe conditional structures.

Single alternative

This structure has the form

If condition then,[Module
A][End of if structure]
Here, the if condition holds, then module A which consists of one or more statements is executed; otherwise Module A is skipped and control transfers to the next step of the algorithm.

Double alternative
This structure has the form If condition, then:
[Module A]
Else
[Module B]
[End of if structure]
The logic of this structure is picture in fig. 2-4(b). As indicated by the flowchart, if the condition holds, then module A is executed; otherwise Module B is executed.

Multiple alternative
This structure has the form:
If condition(1), then:
[Module A1]
Else if condition(2), then:
[Module A2]
.
.
.
Else if condition (M), then:
[Module AM] Else:
[Module B]
[End of if structure]

The logic of this structure allows only one of the modules to be executed. Specifically, either the module which follows the first condition which holds is executed, or the module which follows thefinal Else statement is executed. In practice, there will rarely be more than three alternatives.

2. Describe loop structures.

The repeat-for loop uses an index variable, such as K to control the loop. The loop will usually have the form:

```
Repeat for k=R to S by T:
  [Module]
[End of loop]
```

The repeat-while loop uses a condition to control the loop. The loop will usually have the form

```
Repeat while condition:
[module]
[end of
loop]
```

The looping continues until the condition is false

3. What do you mean by a recursion? Give any 2 recursive techniques.

Recursion is the name given to the technique of defining a set or a process in terms of itself. A recursive procedure can be called from within or outside itself.

Suppose P is a procedure containing either a call statement to itself or a call statement to a second procedure that may eventually result in a call statement back to the original procedure P. then P is called a recursive procedure.

A recursive procedure must have the following 2 properties.

1. There must be certain criteria, called base criteria, for which the procedure does not call itself.
2. Each time a procedure calls itself, it must be closer to the base criteria.

A recursive procedure with these two properties is said to be well-defined.
Examples: Factorial function
Fibonacci function
Towers of Hanoi

4. Explain Factorial function by using Recursive technique.

The product of positive integers from 1 to n, is called" n factorial" and is usually denoted

by n!N! =1 2 3..(N-1) N

Where 0! =1 always. Therefore, for every positive integer n, n! =n(n-1)!
Accordingly, the factorial function may also be defined as follows:

a) If n=0, then n! =1
b) If n>0, the n! =n.(n-1)!

Observe that the definition of n! is recursive, since it refers to itself when it uses (n-1)!However

1. The value of n! is explicitly given when n=0 (thus 0 is the base value)
2. The value of n! for arbitrary n is defined in terms of a smaller value of n which is closer to the base value 0 accordingly, this definition is not circular. And is well defined.
The following procedures calculate n factorial.

<u>Procedure Factorial (Fact, n)</u>

This procedure calculates n! and returns the value in the variable Fact.

```
1. If n=0 then
2. Set Fact: =1 and return
3. Call Factorial (Fact, n-1)
4. Set Fact: =n*Fact(n-1)
5. Return
```

5. Explain Fibonacci series by using recursive technique.

The Fibonacci sequence is as follows 0,1,1,2,3,5,8,13
That is $F_0=0$ and $F_1=1$ and each succeeding term is the sum of two preceding terms. A formal definition of this function is as follows:

a) If n=0 or n=1, then $F_n=n$
b) If n>1, then $F_n=F_{n-2}+F_{n-1}$

This is another example of a recursive definition, since the definition refers toitself when it uses F_{n-2} and F_{n-1}. Here

a) The base values are 0 and 1
b) The value of F_n is defined in terms of smaller values of n which are closer to the basevalues. Accordingly, this function is well defined.

A procedure for finding the nth term F_n of the Fibonacci sequence follows:

<u>Procedure FIBONACCI (Fib, n)</u>

This procedure calculates Fn and returns the value in the first parameter Fib.

```
1. If n=0 or n=1 then
2. Set Fib=n and return
3. Call Fibonacci(Fib a, n-2)
4. Call Fibonacci(Fib b, n-1)
5. Set Fib:=Fib a+Fib b
6. Return
```

6.Explain Towers of Hanoi by using recursive technique.

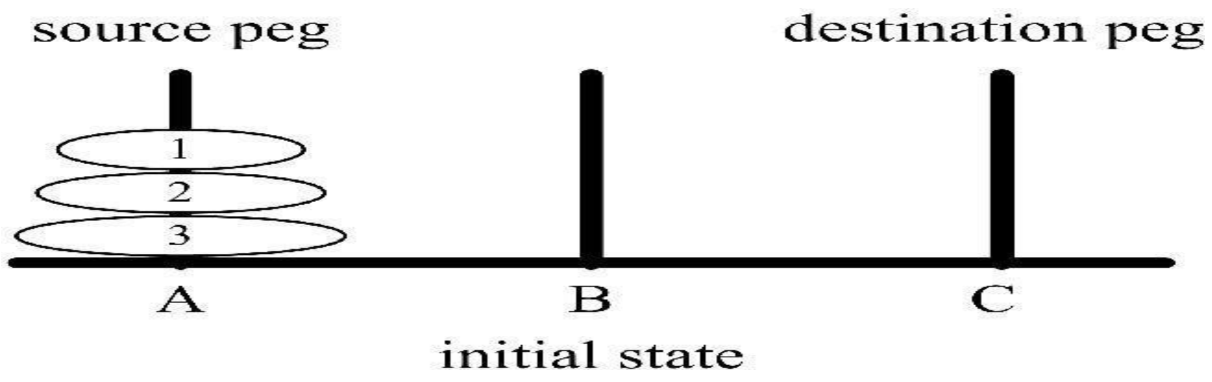
The Tower of Hanoi is a mathematical puzzle invented by the French mathematician Eduard Lucas in 1883.
There are three pegs, source(A), Auxiliary (B), and Destination(C). Peg A contains a set of disks stacked to resemble a tower, with the largest disk at the bottom and the smallest disk at the top. figure 1 Illustrate the initial configuration of the pegs for 3 disks. The objective is to transfer the entire tower of disks in peg A to peg C maintaining the same order as the disks.

Rules:

1. Only one disk can be transferred at a time.
2. Each move consists of taking the upper disk from one of the pegs and placing it on thetop of another peg i.e. a disk can only be moved if it is the uppermost disk of the peg.
3. Never a larger disk is placed on a smaller disk during the transfer.

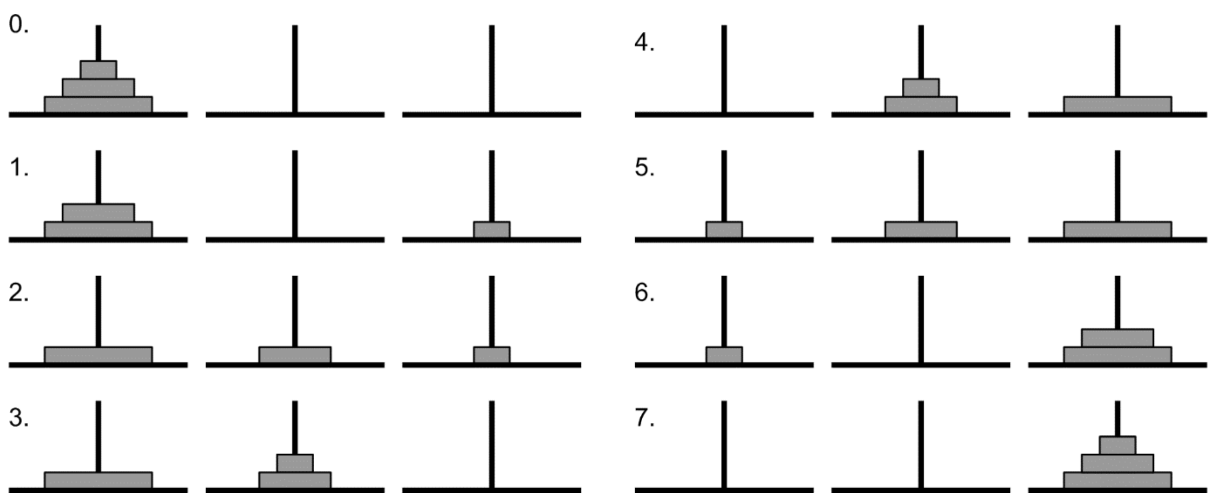
A recursive procedure for the solution of the problem for N numberof disks is as follows:

1. Move the top N-1 disks from peg A to peg B (using C as an auxiliary peg)
2. Move the bottom disk from peg A to peg C
3. Move N-1 disks from Peg B to Peg C (using Peg A as an auxiliary peg)



Algorithm:

```
TOH( n, Sour, Aux , Des)
If(n=1)
    Write ("Move Disk ", n ," from ", Sour ," to ",Des)Else
TOH(n-1,Sour,Des,Aux);
    Write ("Move Disk ", n ," from ", Sour ," to ",Des)
TOH(n-1,Aux,Sour,Des);
END
```



MODULE 2

Multiple Choice Questions

(Questions for Understanding)

1. Full form of access is used to add and remove nodes from a queue.
A.LIFO
B.**FIFO**
C.Both a and b
D.None of these

2. In the linked representation of the stack behaves as the top pointer variable of the stack.
A. Stop pointer
B. Begin pointer
C. **Start pointer**
D. Avail pointer

3. New nodes are added to the queue in______position.
A. Front
B. **Rear**
C. Middle
D. Both A and B

4. In the linked representation of the stack the null pointer of the last node in the list indicates
A. Beginning of the stack
B. **Bottom of the stack**
C. Middle of the stack
D. In between some value

5. What happens when you push a new node onto a stack?
A. **The new node is placed at the front of the linked list**
B. The new node is placed at the back of the linked list
C. The new node is placed in the middle of the linked list
D. No Changes happen

6. A queue is a data structure.
A. **FIFO**
B. LIFO
C. FILO
D. LOFI

7. The term push and pop is related to
A. Array
B. Lists
C. **Stacks**
D. Trees

8. Which is the pointer associated with the stack?
A. FIRST
B. FRONT
C. **TOP**
D. REAR

9. The elements are removed from a stack in order.
A. **Reverse**
B. Hierarchical
C. Alternative
D. Sequential

10. The insertion operation in the stack is called_
A. Insert
B. **Push**
C. Pop
D. top

11. Is the term used to insert an element into the stack.
A. **Push**
B. Pull
C. Pop
D. Pump

12. Stack follows the strategy of
A. **LIFO**
B. FIFO
C. LRU
D. RANDOM

13. is the term used to delete an element from the stack.
A. Push
B. Pull
C. **Pop**
D. Pump

14. Deletion operation is done using in a queue.
A. **front**
B. rear
C. top
D. list

15. A pointer variable that contains the location at the top element of the stack is called ______
A. **Top**
B. Last
C. Final
D. End

16. Which of the following is an application of stack?
A. finding factorial
B. tower of Hanoi
C. infix to postfix
D. **all of the above**

17. Circular Queue is also known as
A. **Ring Buffer**
B. Square Buffer
C. Rectangle Buffer
D. Curve Buffer

18. A data structure in which elements can be inserted or deleted at/from both ends but not in the middle is?
A. Queue
B. Circular queue
C. **Dequeue**
D. Priority Queue

(Questions for Skill)

19. In the linked representations of the stack holds the elements of the stack.
A. **INFO fields**
B. TOP fields
C. LINK fields
D. NULL fields

20. ___________ is a form access is used to add or remove nodes from a stack.
A. LIFO
B. FIFO
C. Both A and B
D. None of these

21. Which of the following name does not related to stack?
A. **FIFO**
B. LIFO
C. Piles
D. Push down lists

22. The retrieval of items in a stack Operation is.
A. push
B. **pop**
C. retrieval
D. access

23. The expression +a*bc is in notation.
A. Infix
B. **Prefix**
C. Postfix
D. Reverse Polish

24. The expression a+b*c is in notation.
A. **Infix**
B. Postfix
C. Prefix
D. Reverse polish

25. If the elements "A"," B","C" and "D" are placed in a queue and are deleted one at a time, in what order will they be removed?
A. **ABCD**
B. DCBA
C. DCAB
D. ABDC

Questions carrying 4 marks

(Questions for Understanding)

1. **What do you mean by a queue? What are operations involved in a queue?**
A queue is a linear list of elements in which deletion takes place at only one end called the front and insertion takes place at another end called the rear. It is also called FIFO data structure, the First in first out data structure. It contains two pointer variables FRONT and REAR where the front point is to the front element in the queue and the rear point is to the rear element in the queue.
Operations involved in a queue are:
1) **Insertion**: whenever an element is added to the queue, the value of REAR is increased by 1.
REAR: =REAR + 1;
2) **Deletion**: Whenever an element is deleted from the queue the value of the front is increased by 1.
FRONT: = FRONT + 1;

2. **Give any four applications of a stack.**
   - Evaluation of Arithmetic Expressions: A stack is a very effective data structure for evaluating arithmetic expressions in programming languages. Anarithmetic expression consists of operands and operators. In addition to operands and operators, the arithmeticexpression may also include parenthesis like "left parenthesis" and "right parenthesis".
     Example: A + (B - C) Notations for Arithmetic Expression
   - Backtracking: Backtracking is another application of Stack. It is a recursive algorithm that is used for solving the optimization problem.
   - Delimiter Checking: The common application of Stack is delimiter checking, i.e., parsing that involves analyzing a source program syntactically. Itis also called parenthesis checking. We make use of different types of delimiters include the parenthesis checking (,), curly braces {,} and square brackets [,], and common delimiters /* and */.
   - Reverse a Data: To reverse a given set of data, we need to reorder the data so that the first and last elements are exchanged, the second and second last element are exchanged, and so on for all other elements.

3. **Give any four applications of a queue.**
   - When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling.
   - When data is transferred asynchronously (data not necessarily received at the same rate assent) between twoprocesses. Examples include IO Buffers, pipes, etc
   - Semaphores
   - FCFS (first come first serve) scheduling, for example, FIFO queue

- Spooling in printers
- Buffer for devices like keyboards.
- Queues in routers/ switches
- Mail Queues
- Variations: (Deque, Priority Queue, Doubly Ended Priority Queue).
- Applied as waiting lists for a single shared resource like CPU, Disk, and Printer.
- Applied as buffers on MP3 players and portable CD players.
- Applied on Operating system to handle the interruption.
- Applied to add a song at the end or to play from the front.

4. **Write the algorithm for the linked list representation of a stack.**

To implement a stack using the singly linked list concept, all the singly linked list operations should be performed based on Stack operations LIFO(last in first out) and with the help of that knowledge, we are going to implement a stack using a singly linked list.
So we need to follow a simple rule in the implementation of a stack which is last in first out and all the operations can be performed with the help of a top variable.

Algorithm:

push(value) - Inserting an element into the Stack

We can use the following steps to insert a new node into the stack.

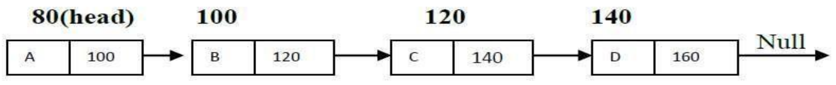
- Step 1 - Create a newNode with given value.
- Step 2 - Check whether stack is Empty (top == NULL)
- Step 3 - If it is Empty, then set newNode → next = NULL.
- Step 4 - If it is Not Empty, then set newNode → next = top.
- Step 5 - Finally, set top = newNode.

pop() - Deleting an Element from a Stack

We can use the following steps to delete a node from the stack.

- Step 1 - Check whether stack is Empty (top == NULL).
- Step 2 - If it is Empty, then display "Stack is Empty!!! Deletion is not possible!!!" and terminate the function
- Step 3 - If it is Not Empty, then define a Node pointer 'temp' and set it to 'top'.
- Step 4 - Then set 'top = top → next'.
- Step 5 - Finally, delete 'temp'. (free(temp)).

5. **Write the algorithm for the linked list representation of a queue.**

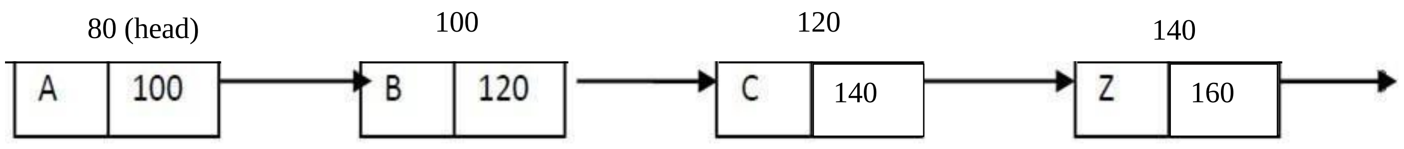


- Two pointer variables FRONT and REAR pointing to the nodes which is in the front and rear of the queue.
- The INFO field of the node holds the data in the queue.
- The NEXT is the pointer to the next element in the queue.

Procedure LQINSERT(INFO,NEXT, ITEM, REAR,FRONT)

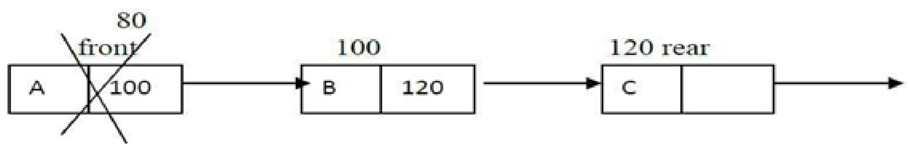
This procedure inserts an element ITEM into a linked queue.

1. N= new node
2. If (n==NULL)
3. Then write (“Overflow”) and exit
4. N->info=ITEM
5. If (Front==NULL) then
6. Front=Rear=n
7. Else
8. Rear->next=n
9. Rear=n
10. N->next=NULL
11. finished



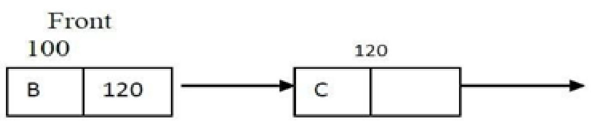
Procedure LQDELETE(INFO, NEXT,FRONT, REAR, ITEM,TEMP)

This procedure deletes the front element of the linked queue and stores it in ITEM.



1. Temp= front
2. Item=front->info
3. Front=temp->next
4. Delete temp
5. Finished

AFTER DELETION:



6. **Write a note on the priority queue and dequeue.**

Priority queue: This is a collection of elements such that each element has been assigned a priority and such that the order in which elements are deleted and processed takes place. An element of higher priority is processed before any element of lower priority. Two elements with the same priority are processed according to the order in which they were added to the queue.

Dequeue: Double-ended queue in which elements can be added or removed at either end but not in the middle. Two variations:-

1) Input restricted deque: An input-restricted deque is a deque that allows insertions at only one end of the listbut allows deletions at both ends of the list.

2) Output restricted deque: An output-restricted deque is a deque that allows deletions at only one end of the list butallows insertions at both ends of the list.

(Questions for Skill)

1. **Write an algorithm to insert and delete elements from/to a circular queue.**

PROCEDURE CQINSERT(F,R,Q,N,Y): Given F and R as the pointers to the front and rear element of a circular queue. The array Q contains N elements. Y is the element to be inserted at the rear.

```
Step 1: IF (REAR+1)%MAX = FRONT
Write " OVERFLOW "
Goto step 4
[End OF IF]
Step 2: IF FRONT = -1 and REAR = -1
SET FRONT = REAR = 0
Write " OVERFLOW "
ELSE IF REAR = MAX - 1 and FRONT! = 0
SET REAR = 0
ELSE
SET REAR = (REAR + 1) % MAX
[END OF IF]
Step 3: SET QUEUE[REAR] = VAL
Step 4: EXIT
```

FUNCTION CQDELETE(F,R,Q,N): Given F and R as the pointers to the front and rear element of a circular queue. The array Q contains N elements. This function deletes and returns the last element of the queue. Y is the temporary variable.

```
Step 1: IF FRONT = -1
Write " UNDERFLOW "
Goto Step 4
Step 2: SET VAL = QUEUE[FRONT]
Step 3: IF FRONT = REAR
SET FRONT = REAR = -1
ELSE
IF FRONT = MAX -1
```

```
SET FRONT = 0
ELSE
SET FRONT = (FRONT + 1)%N
[END of IF]
[END OF IF]
Step 4: EXIT
```

2. **Evaluate ABC*+D- with proper step. Assume A=4, B=6, C=2, D=4.**

ABC*+D-
462*+4-

Step	Input Symbol/Element	Stack	Intermediate Output
1	4 Push	4	
2	6 Push	4 6	
3	2 Push	4 6 2	
4	* Pop 2 elements and evaluate	4	12
5	Push result 12	4 12	16
6	+ Pop 2 elements and evaluate	16	
7	Push result 16	16 4	12
8	4 Push		
9	- Pop 2 elements and evaluate	12	**12**
10	Push result 12		
11	No more elements		

3. **Write an algorithm to convert infix expressions to postfix expressions.**

```
1. Push "("onto Stack, and add ")" to the end of X.
2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.
3. If an operand is encountered, add it to Y.
4. If a left parenthesis is encountered, push it onto Stack.
5. If an operator is encountered, then:
      1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the
         same precedence as or higher precedence than operator.
      2. Add operator to Stack.
   [End of If]
6. If a right parenthesis is encountered, then:
      1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) until a
         left parenthesis is encountered.
      2. Remove the left Parenthesis.
   [End of If]
[End of If]
7.END
```

OR

Procedure Postfix (inputExpr, St, Symbol, OutputExpr)

This procedure converts an infix expression to postfix expression. The infix expression is taken in *inputExpr*, *st* represents a new stack, *Symbol* is the current Symbol, *stack_top_symbol* is the symbol at the top of the stack, t*opSymbo*l is/are the symbol(s) remained in the stack at the end of the evaluation and *outputExpr* is the postfix expression

```
St= new stack
Repeat for each character in inputExpr
Begin
Symbol=current character in inputExpr
if (symbol is an operand)
add symbol to outputExpr
else
begin
while( not st.empty && (priority(symbolstack_top_symbol)))
Begin
Add stack_top_symbol to outputExpr
End while
St.push(symbol)
End else
End for
While(not st.empty)
Begin

/*to pop the remaining operators from the stack at the end.*/
topSymbol=st.pop()
add topSymbol to outputExpr
end while
```

4. Write an algorithm to evaluate postfix expressions.

```
1. Add a right parenthesis ")" at the end of P.
2. Scan P from left to right and repeat Steps 3 and 4 for each element of P until the ")" is encountered.
3. If an operand is encountered, put it into a stack.
4. If an operator is encountered then
Remove the two top elements of the Stack, where A is the top element and B is the next-top-top element.
b) Evaluate B operator A
c)Place the result of b) back on Stack
[End of If structure]
[End of Step 2 loop]
A set value equal to the top element on Stack.
5.Exit
```

OR

Consider the following expression ABC+* and assume that A is 2, B is 4 and C is 6. Here you should notice that the operands come before the operators. Whenever we find an operator, we put them in the stack. When we find an operator, we pop the 2 operands at the top of the stack and perform the operation. We then push the value back into the stack to be used later.

FUNCTION Evaluate_Postfix(Expression,st, symbol)

This procedure evaluates an postfix expression. Expression is the postfix expression. st represents a new stack, Symbol is the current symbol

```
St=new stack
Repeat for every character in the expression
Begin
Symbol=current character in the expression
If(symbol is an operand)
St.push(symbol)
END
Else
Begin
Operand2=st.pop()
Operand1=st.pop()
Answer=operand2 and operand1 operated with symbol
St.push(answer)
END
Return st.pop()//display the output
```

5. **Write the PUSH and POP operations in a stack.**

PROCEDURE PUSH(S, TOP, X)

This procedure inserts an element X into the top of the stack which is represented byan array S. The array S contains N elements with a pointer TOP denoting the top element inthe stack.

```
1. [Check for Overflow]
   If TOP>=N
        Then write (“OVERFLOW”)
   Return
2. [Increment TOP]
   TOP=TOP+1
3. [Insert Element]
   S[TOP]=X
4. [Finished]
   Return
```

FUNCTION POP(S,TOP):

This function removes the top element from a stack which is represented by a vector Sand returns this element. TOP is a pointer to the top element of the stack.

```
1.[Check for underflow on stack]If TOP=0
  Then
  write(“stack underflow)

  Exit
2.[Decrement TOP pointer]TOP=TOP-1
3.[Return former top element of the Stack]Return
(S[TOP+1])
```

6. **Evaluate 2 3 1 * + 9 - the postfix expression with proper steps.**

2 3 1 * + 9 –

Step	Input Symbol/Element	Stack	Intermediate Output
1	2 Push	2	
2	3 Push	2 3	
3	1 Push	2 3 1	
4	* Pop 2 elements and evaluate	2	3
5	Push result 3	2 3	5
6	+ Pop 2 elements and evaluate		
7	Push result 5	5	
8	9 Push	5 9	4
9	- Pop 2 elements and evaluate		
10	Push result 4	4	**4**
11	No more elements		

MODULE 3

Multiple Choice Questions

(Questions for Understanding)

1. Each node in a doubly linked list consists of_____________________fields.
A. One
B. Two
C. Three
D. Four

2. A linked list is an example of______________________data structure.
A. Static
B. Dynamic
C. Heterogeneous
D. Sequential

3. The data field in a node of a linked list represents __________________
A. Address of the data
B. Link to the next node
C. Value to be processed
D. Memory location of the node

4. Which of the following is two-way lists?
A. Grounded header list
B. Circular header list
C. Linked list with header and trailer nodes
D. List traversed in two directions

5. Which of the following is not a disadvantage to the usage of array?
A. Fixed size
B. There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size
C. Insertion based on position
D. Accessing elements at specified positions

6. In linked lists there are no NULL links in
A. single linked list
B. linear doubly linked list
C. circular linked list
D. linked list

7. Each node in a linked list must contain at least_________________
A. Three fields
B. Two fields
C. Four fields
D. Five fields

8. The dummy header in the linked list contain_______________

A. <u>First record of the actual data</u>
B. last record of the actual data
C. pointer to the last record of the actual data
D. middle record of the actual data

9. In a linked list the________________field contains the address of the next element in the list.
A. <u>Link field</u>
B. Next element field
C. Start field
D. Info field

10. LINK is the pointer pointing to the_____________________
A. Successor node
B. <u>predecessor node</u>
C. head node
D. last node

11. ______refers to a linear collection of data items.
A. <u>List</u>
B. Tree
C. Graph
D. Edge

12. A run list is______________________
A. <u>small batches of records from a file</u>
B. number of elements having same value
C. number of records
D. number of files in external storage

13. A linear list in which the pointer points only to the successive node is _______________
A. <u>singly linked list</u>
B. circular linked list
C. doubly linked list
D. none of the above

14. A linear list in which the last node points to the first node is _________________
A. singly linked list
B. **<u>circular linked list</u>**
C. doubly linked list
D. none of the above

15. In circular linked list, insertion of node requires modification of?
A. One pointer
B. <u>Two pointer</u>
C. Three pointer

D. None of these

(Questions for Application)

16. Linked lists are best suited ________________
A. For relatively permanent collections of data.
B. For the size of the structure and the data in the structure are constantly changing.
C. data structure
D. for none of the above situation

17. The operation of processing each element in the list is known as _______________
A. sorting
B. merging
C. inserting
D. traversal

18. The situation when in a linked list START=NULL is _________________
A. Underflow
B. Overflow
C. Houseful
D. Saturated

19. Each node in singly linked list has__________fields.
A. 2
B. 3
C. 1
D. 4

20. Which is the pointer associated with the availability list?
A. FIRST
B. AVAIL
C. TOP
D. REAR

21. Value of first linked list index is ________________
A. 0
B. 1
C. -1
D. 2

22.A_____indicates the end of the list.
A. Guard
B. **Sentinel**
C. End pointer
D. Last pointer

23.A________________is a linear list in which insertions and deletions are made to from either end of the structure.
A. Circular queue
B. random of queue
C. priority
D. **<u>dequeue</u>**

24. Indexing the________________element in the list is not possible in linked lists.
A. <u>middle</u>
B. first
C. last
D. anywhere in between

25. __________may take place only when there is some minimum amount (or) no space left in free storage list.
A.Memory management
B.<u>Garbage collection</u>
C.Recycle bin
D.Memory management

Questions carrying 4 marks

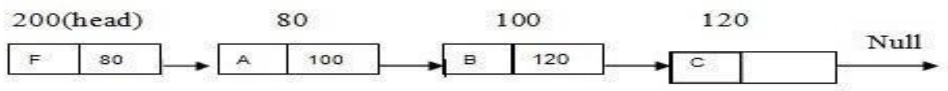
(Questions for Understanding)

1. What do you mean by linked list? With a neat diagram explain different types of linkedlists.

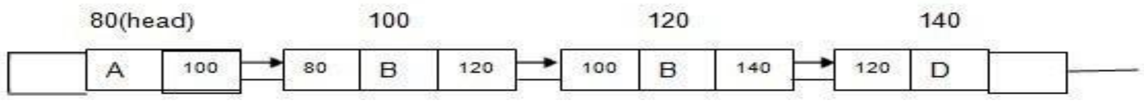
A linked list is a non-sequential collection of data items. For every data item in the linked list, there is an associated pointer that gives the memory location of the next data item in the linked list. The data items in the linked list are not in a consecutive memory locations. But they may be anywhere in memory. However, accessing of these items is easier as each data itemcontained within itself the address of the next data item.

Basically we can put linked lists into following 4 types:-
- Singly linked list
- Doubly linked list
- Singly circular linked list
- Doubly circular linked list

Singly linked list
A singly linked list is the one in which all nodes are linked together in some sequential manner. Hence it is also called linear linked list. It has the beginning and the end.

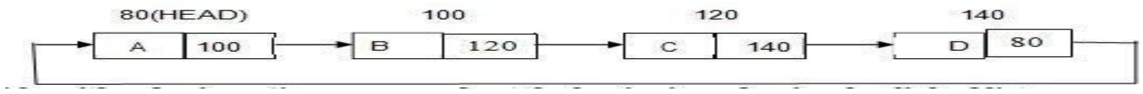


Doubly linked list
A doubly linked list is a one in which all nodes are linked together by multiple number of links which help in accessing both the successor node and the predecessor node form the given node position.



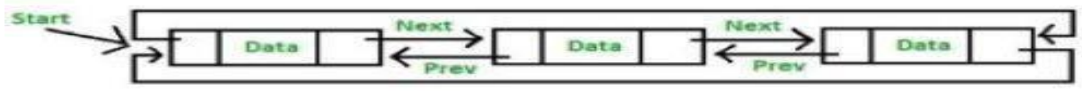
Singly circular linked list

It is just a singly linked list in which the link field f the last node contains the address of the first node of the list.

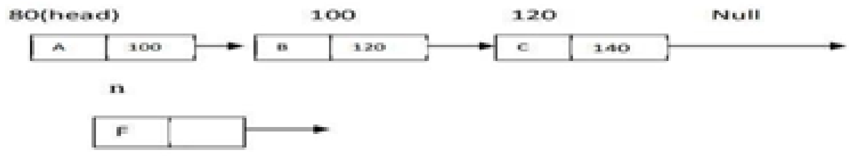


Doubly circular linked list

In circular doubly linked list two consecutive elements are linked or connected by previous and next pointer and the last node points to the first node also points to last node by previous pointer.



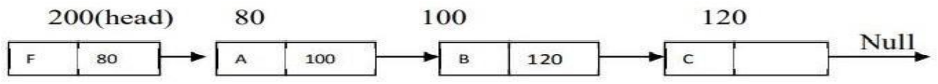
2. **Write an algorithm to insert nodes from the beginning and end, of a singly linked list.**
    - ***Algorithm for inserting a node at the beginning of a linked list***



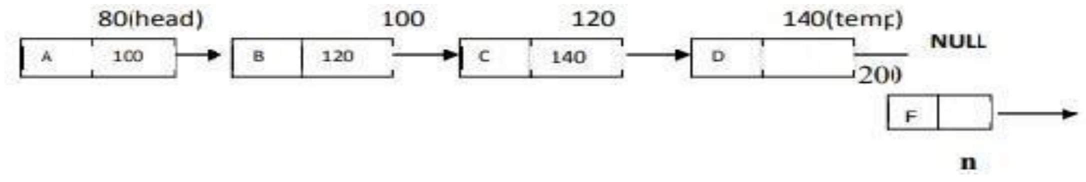
Procedure Insert_Head(n)

This algorithm inserts a new node called n to the beginning of a linked list. Head denote the starting node of the linked list.

1. [make head as the next element to the newnode]
   N->next=head
2. [make our new node as head]
   Head=n
3. [finished]
   Exit



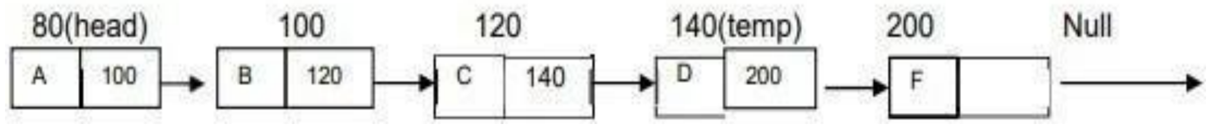
- ***Algorithm for inserting a node at the end of a linked list***



This algorithm inserts a new node called n at the end of the linked list. Temp is a temporary variable where the head node is stored temporarily before insertion takes place.

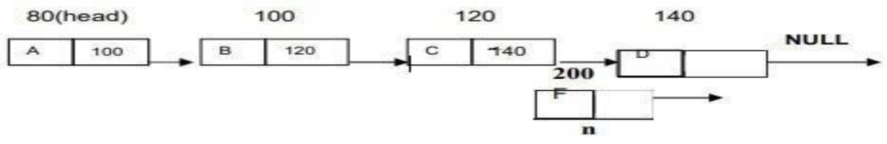
Procedure_Insert_end(n)

```
Step 1: temp=head
Step 2: while(temp->next!=NULL)
begin
Step 3: temp=temp->next
End
Step 4: temp->next=n
Step 5: n-> next=NULL
Step 6: Exit
```



3. Write an algorithm to insert nodes from the middle(specified position), of a singly linked list.

- ***Algorithm for inserting a node in the middle of a linked list***



Procedure insert_middle(n)

This algorithm inserts a new node called n in the specified position of the linked list. After is a node where the new node is to be inserted after it.

1. N $\rightarrow$ next=after $\rightarrow$ next
2. After $\rightarrow$ next =n
3. Exit

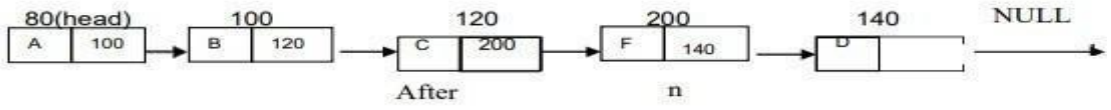
OR

1.Newnode=n

2. newnode $\rightarrow$ next=temp $\rightarrow$ next

3.temp $\rightarrow$ next=newnode

4.Exit



4. Write an algorithm to search for element in a linked list.

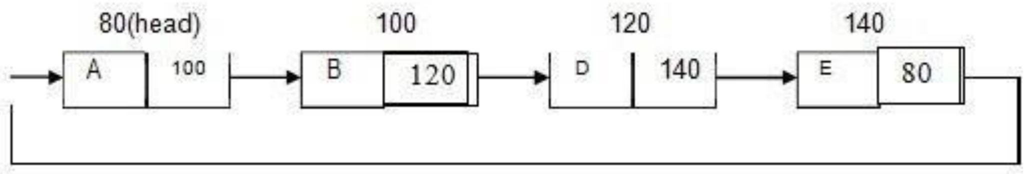
Let List be a linked list in memory. INFO is the pointer pointing to the Information part of a node and NEXT is the pointer which contains the address of the next node in the linked list. HEAD is the starting node. Suppose a specific ITEM of information is given .This algorithm is used for finding the location LOCATION of the node where ITEM first appears in the LIST.

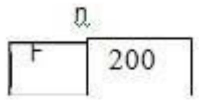
SEARCH(INFO, LINK, HEAD, ITEM, LOCATION)

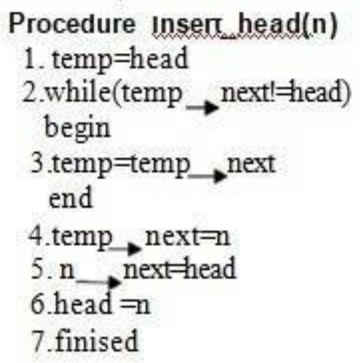
```
Step1 : set temp=head
Step 2: while(temp->next!=NULL)
Begin
Step 3: if item=temp->info
Step 4: set location =temp and
exit
Step 5: else
Step 6: set temp=temp->next
End if
End while
Step 7: set location=NULL
Step 8: exit
```

5. **Write an algorithm to insert nodes from the beginning, end of a circular linked list.**

Algorithm for inserting a new node at the beginning of a circular linked list

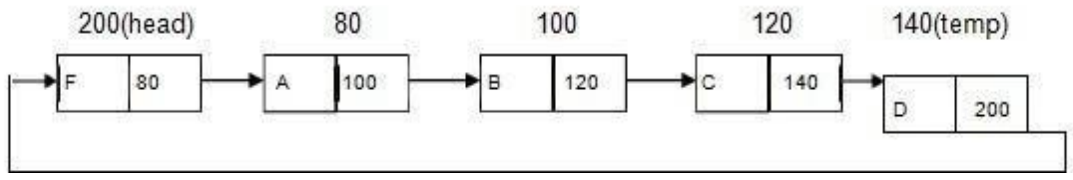




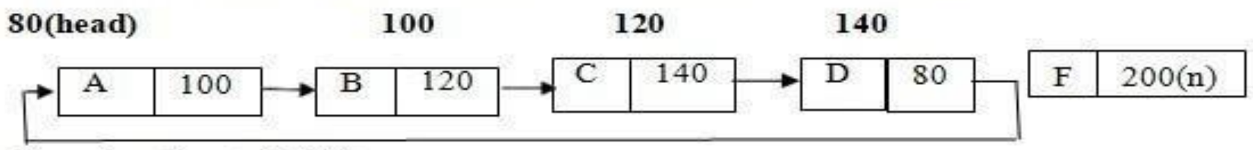


Procedure Insert_head(n)

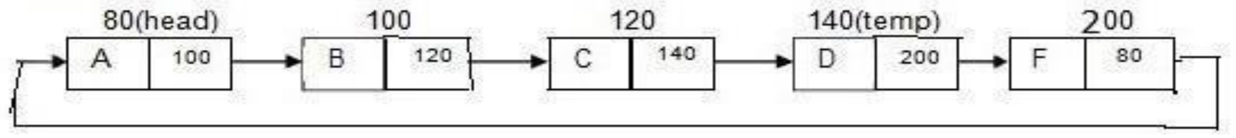
```
1. temp=head
2.while(temp→next!=head)
  begin
 3.temp=temp→next
  end
 4.temp→next=n
 5. n→next=head
 6.head =n
 7.finised
```



Algorithm for inserting a new node at the end of circular linked list



```
Procedure Insert_End(n)
    1. temp=head
2.  while(temp→next!=head)
    begin
3.  temp=temp→next
    end
4.  temp→next=n
5.  n→next=head
6.  finished
```



(Questions for Application)

1. Describe different functions in dynamic memory allocation.

- **malloc()**:- Allocate request size of bytes & return a pointer to the first byte of the allocated space. And contains garbage values.

  Syntax:

  ptr = (cast-type*) malloc(byte-size);

  For Example:

  ptr = (int*) malloc(100 * sizeof(int));

- **calloc()**:- Allocate space for an array of elements, initialize them to zero and returns a pointer to the first byte of allocated space.

  Syntax:

  ptr = (cast-type*)calloc(n, element-size);

  Here, n is the no. of elements and element-size is the size of each element.

  For Example:

  ptr = (float*) calloc(25, sizeof(float));

- **realloc()**:- Modify the size of previously allocated space.

  Syntax:

  ptr = realloc(ptr, newSize);

  where ptr is reallocated with new size 'newSize'.

- **free()**:- Free the previously allocated space.

  Syntax:

  free(ptr);

2. Write an algorithm to traverse a linked list.

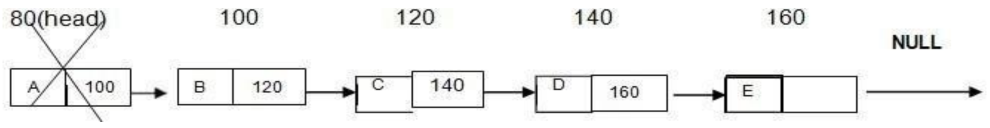
Procedure TRAVERSE (LIST, TEMP,INFO, HEAD, NEXT)

Let LIST be a linked list in memory. This algorithm traverses LIST, applying an operation DISPLAY to each element of LIST. The variable TEMP points to the node which is currently being processed.

```
1. Set Temp=head
2. While(temp!=NULL)
3. Begin
4. Display temp→info
5. Set temp=temp→next
6. End
7. Finished
```

3. Write an algorithm to delete nodes from the beginning and end of a singly linked list.

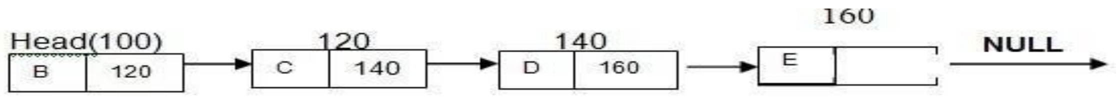
- ***Algorithm for deleting the last node form the linked list***



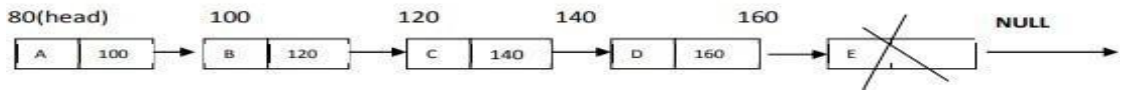
Procedure Delete_Head ()

This algorithm deletes the head node of the linked list. Temp is a temporary variablewhere the head node is stored temporarily before deletion takes place.

```
1. temp=head
2. Head=head→next
3. Delete temp
4. Exit
```



- ***Algorithm for deleting the last node form the linked list***



Procedure delete_end()

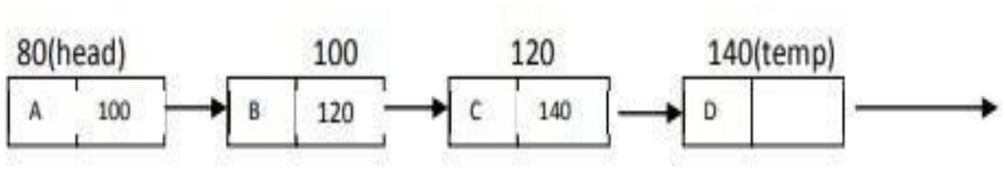
This algorithm deletes the last node of the linked list. Temp is a temporary variablewhere the head node is stored temporarily before deletion process takes place.

```
1. Temp=head
2. While(temp->next->next!=NULL)
3. Begin
4. Temp=temp->next
5. End
6. Delete temp->next
```

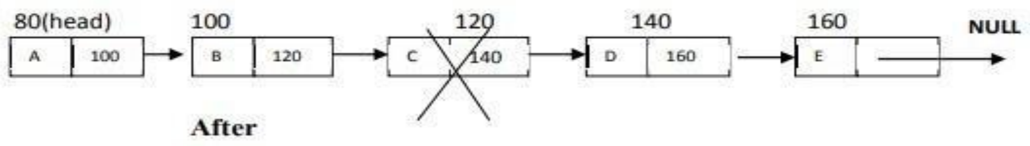
```
7. Temp->next=NULL
8. Exit
```

OR

```
1. Temp=head
2. While(temp->next!=NULL)
   Begin
3.    Previousnode=temp
4.    Temp=temp->next
   End
5. Previousnode->next=NULL
6. Delete Temp
```



4. Write an algorithm to delete nodes from the specified position(middle) of a singly linked list.



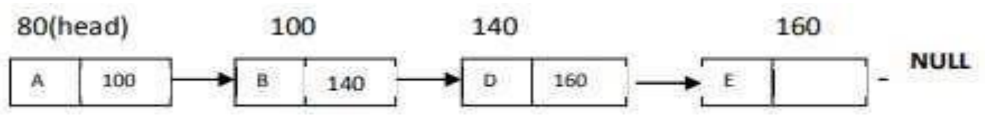
Procedure Delete_middle(n)

This algorithm deletes a node from the middle of the linked list. Temp is a temporary variable where the node is stored temporarily before deletion takes place. After is a node where the node to be deleted after that.

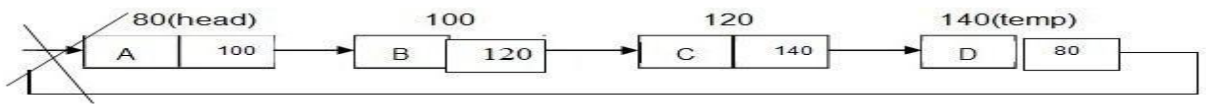
```
1. Temp=after→next
2. After→next=after→next→next
3. Delete temp
4. Exit
```

OR

```
1. nextnode=temp->next
2. temp->next=nextnode ->next
3. Delete nextnode
4. Exit
```

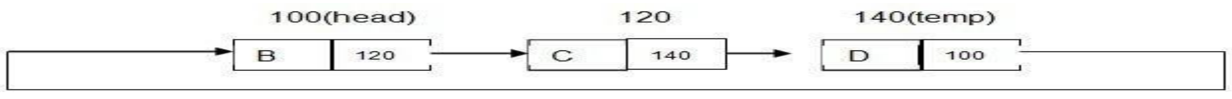


5. Write an algorithm to delete nodes from the beginning of a circular linked list.

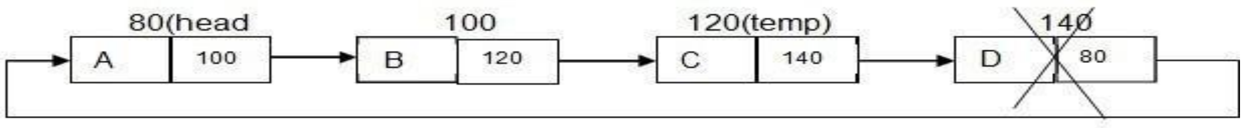


Procedure delete_head()

```
1. temp=head
2. while(temp->next!=head)
3. begin
4. temp=temp→next
5. end
6. Temp→next=head→next
7. Delete head
8. head=temp→next
9. finished
```



6. Write an algorithm to delete nodes from the end of a circular linked list.

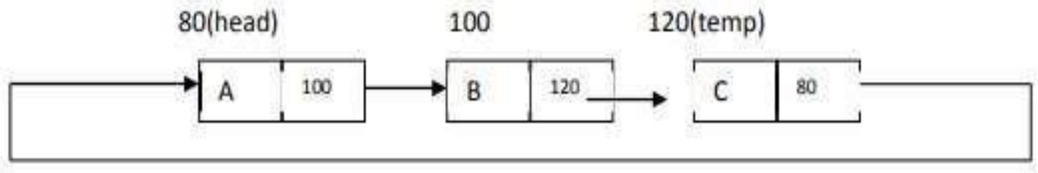


Procedure delete_end()

```
1. Temp=head
2. while(temp->next!=HEAD)
    Begin
3. Previousnode=temp
4. Temp=temp->next
    End
5.  Previousnode->next=HEAD
6. Delete Temp
```

OR

```
1.temp=head
2.while(temp →next→next!=head)
  begin
3. temp=temp→next
  end
4.  delete temp→next
5.  temp→next=head
6.  finished
```



MODULE 4

Multiple Choice Questions

(Questions for Understanding)

1. The number of edges from the node to the deepest leaf is called_______________of the tree.
A. Height
B. Depth
C. Length
D. Width

2. Binary trees with threads are called as______________
A. Threaded trees
B. Pointer trees
C. Special trees
D. Special pointer trees

3. Graph G is_____________if for any pair u, v of nodes in G there is a path from u to v orpath from v to u.
A. Laterally connected
B. Widely Connected
C. **Unilaterally connected**
D. Literally connected

4. In Binary trees nodes with no successor are called______________
A. End nodes
B. Terminal nodes
C. Final nodes
D. Last nodes

5. Trees are said_____________if they are similar and have same contents at corresponding nodes.
A. Duplicate
B. Carbon copy
C. Replica
D. Copies

6. Every node N in a binary tree T except the root has a unique parent called the of N.
A. Antecedents
B. Predecessor
C. Forerunner
D. Precursor

7. Sequential representation of binary tree uses_______________
A. **Array with pointers**
B. Single line array
C. Two dimensional arrays
D. Three dimensional arrays

8. A binary tree whose every node has either zero or two children is called_______________
A. complete binary tree
B. binary search tree
C. extended binary tree
D. data structure

9. In a binary-tree, nodes with 0 children are called___________

A. Exterior node

B. Outside node

C. Outer node

D. External node

10. Which indicates pre- order traversal?

A. Left sub-tree, Right sub-tree and root

B. Right sub-tree, Left sub-tree and root

C. **Root, Left sub-tree, Right sub-tree**

D. Right sub-tree, root, Left sub-tree

12. An edge in a graph with no specific direction is called a ________________________

A. Directed edge

B. Undirected edge

C. Isolated edge

D. Adjacent edge

13. TREE [1] =NULL indicates tree is_______________

A. Overflow

B. Underflow

C. Empty

D. Full

14. Linked representation of binary tree needs_____________parallel arrays.

A. 4

B. 2

C. **3**

D. 5

15. The depth of complete binary tree is given by_______________

A. Dn = nlog2n

B. Dn= nlog2n+1

C. Dn =log2n

D. Dn =log2n+1

16. A terminal node in a binary tree is called_________________.

A. Root

B. Leaf

C. Child

D. Branch

17. Total number of outgoing edges connected to a vertex is said to be______.

A. Outdegree of vertex

B. Indegree of vertex

C. Isolated vertex

D. Internal vertex

18. The number of edges from the root to the node is called _______ of the tree.
A. Height
B. <u>Depth</u>
C. Length
D. Width

19. What is a full binary tree?
A. **<u>Each node has exactly zero or two children</u>**
B. Each node has exactly two children
C. All the leaves are at the same level
D. Each node has exactly one or two children

20. Which of the following is not an advantage of trees?
A. Hierarchical structure
B. Faster search
C. Router algorithms
D. <u>Undo/Redo operations in a notepad</u>

21. What is a threaded binary tree traversal?
A. a binary tree traversal using stacks
B. a binary tree traversal using queues
C. a binary tree traversal using stacks and queues
D. <u>a binary tree traversal without using stacks and queues</u>

22. What are the disadvantages of normal binary tree traversals?
A. **<u>there are many pointers which are null and thus useless</u>**
B. there is no traversal which is efficient
C. complexity in implementing
D. improper traversals

23. Three standards ways of traversing a binary tree T with root R.
A. Prefix, infix, postfix
B. Pre-process, in-process, post-process
C. Pre-traversal, in-traversal, post-traversal
D. <u>Pre-order, in-order, post-order</u>

24. A connected graph T with any cycles is called
A.a tree graph
B.free tree
C.tree
D.**<u>all of the above</u>**

25. Which of the following tree traversals work if the null left pointer pointing to the predecessor and null right pointer pointing to the successor in a binary tree?
A. **<u>inorder, postorder, preorder traversals</u>**
B. inorder
C. postorder
D. preorder

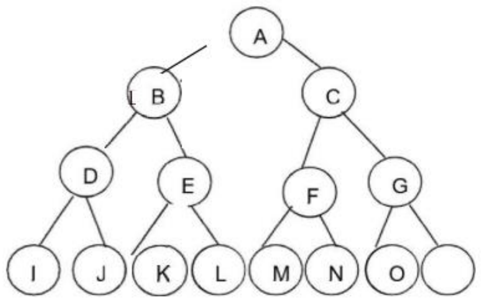
Questions carrying 4 marks

(Questions for Understanding)

1. **What do you mean by the following:**

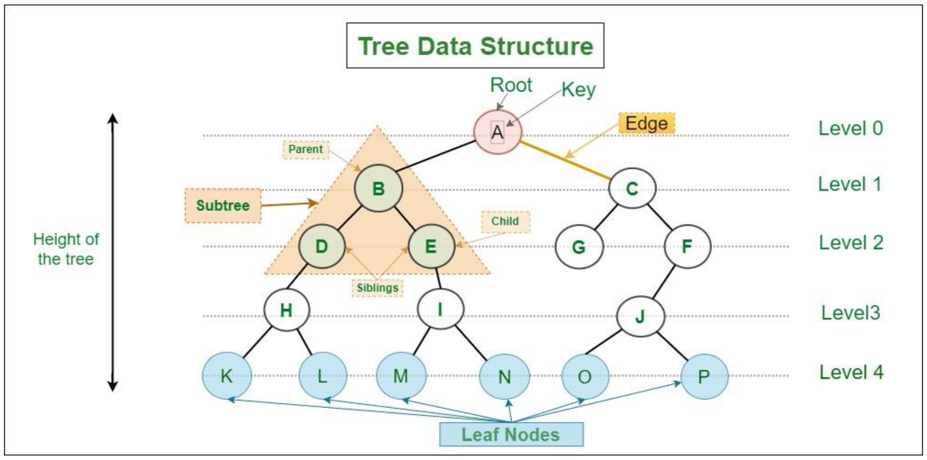
a. Complete binary tree
b. Forest
c. Level of a tree
d. Depth of a tree

a. Complete binary tree:



A complete binary tree with depth 4

In a complete binary tree, there is exactly one node at level 0, 2 nodes at level 1, 4 nodes at level 2 and so on



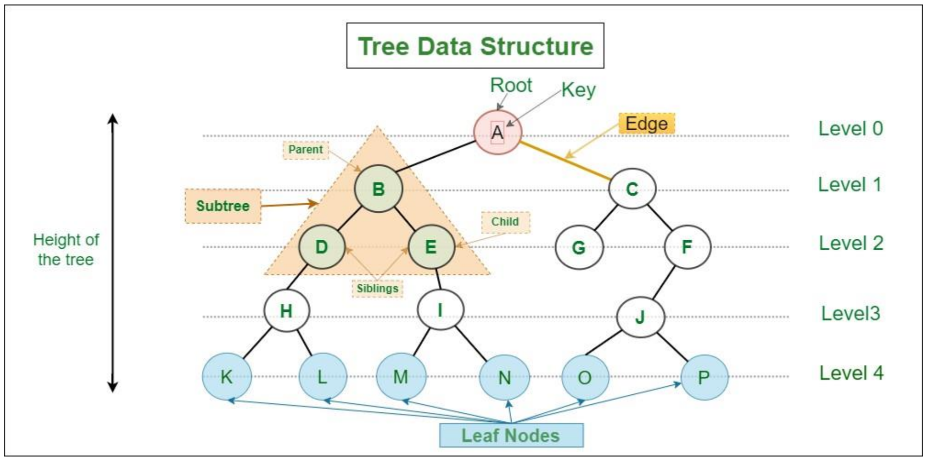
b. Forest:
It is the set of disjoint trees. In a given tree, if you remove its root node then itbecomes a forest. In the above tree, there is a forest with 5 trees.

c. Level:
The entire tree structure is leveled in such a way that the root node is always at level0, then the intermediate children are at level1 and their intermediate children are atlevel2and so on. The above tree, there are 4 levels.

d. Depth:
It is the maximum level of any node in a given tree. In the above tree, the root node Ahas the maximum level. The term height is also used to denote the depth.

2. **What do you mean by the following:**

a. **Degree**
b. **Terminal and Non-Terminal Node**
c. **Path**
d. **Siblings**



a. **Degree of a node**:
It is the number of sub trees of a node in a given tree. In the above tree -
   a) Degree of A is 2
   b) Degree of C is 2
   c) Degree of D is 1
   d) Degree of H is 2
   e) Degree of K is 0

b. **Terminal node**:
A node with degree zero is called a terminal node or a leaf. In the above tree, there are7 terminal nodes. They are K,L ,M,N,G,O and P

Non-terminal Node:
Any node except the root node whose degree is non zero called non-terminalnode, there are 5 non-terminal nodes. They are B,C,D,E,F,H,I and J

c. **Siblings**:
The children nodes of a given parent node are called siblings. They are also called brothers. In the above tree,
   a) D& E are siblings of parent node B.
   b) G & F are siblings of parent node C.

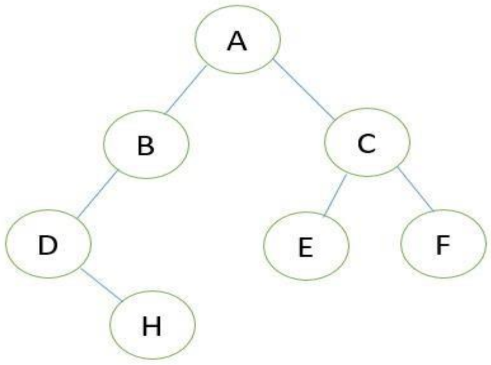
d. **Path**: it is the sequence of consecutive edges from the source node to the destination node. In the above tree, the
path between A and J is given by the node pairs (A,C), (C, F) and(F,J)

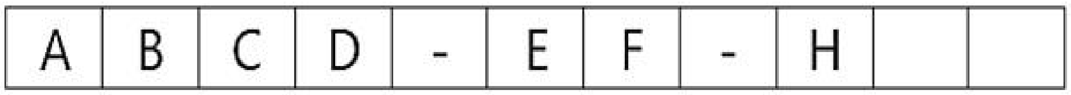
3. **Explain the sequential representation of a binary tree.**

A binary tree contains one root node and some non-terminal and terminal nodes. It is clear from the observation of a binary tree that the non-terminal nodes have their left child nodes. Their lchild and rchild pointer are set to NULL. Here, non-terminal nodes are called internal nodes and terminal nodes are called external nodes.
Sequential representation of Binary trees Suppose T is a binary tree that is complete or nearly

complete. Then there is an efficient way of maintaining T in memory called the sequential representation of T. this representation uses only a single linear array TREE as follows:

1) The root R of T is stored in TREE[1]
2) If a node N occupies TREE[K],then its left child is stored in TREE[2*K] and its right child is stored in TREE[2*K+1]





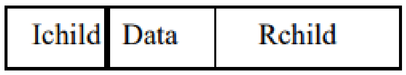
A	B	C	D	-	E	F	-	H		

4. **Explain the linked list representation of a binary tree.**

Binary trees can be represented either using an array representation or using a linked list representation. The basic component to be represented in a binary tree is a node.The node consists of 3 fields such as

1) Data
2) Left child
3) Right child

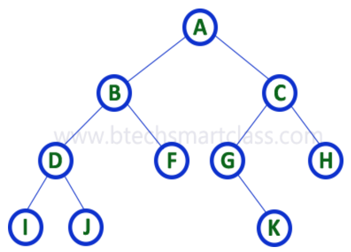
The data filed holds the value to be given. The left child is a link filed which contains the address of its left node and the right child contains the address of its right node.

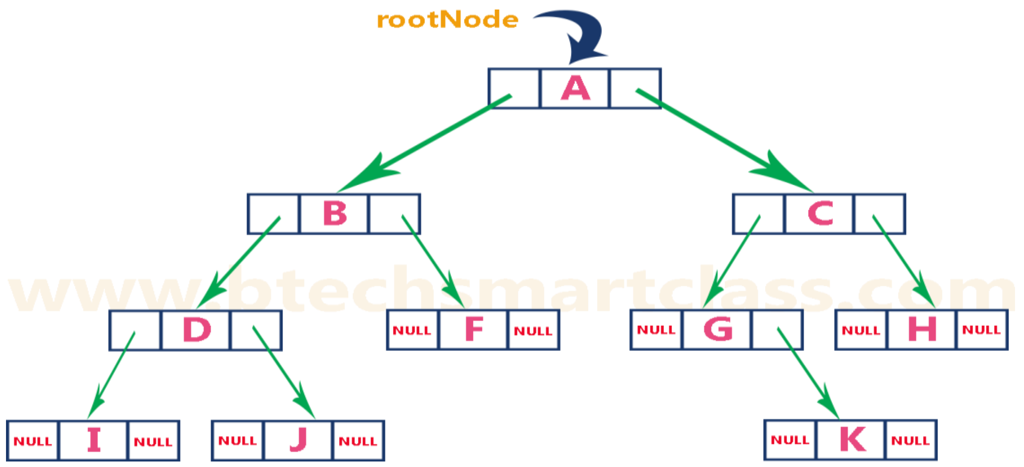


lchild	Data	Rchild

```
class node
{
char data;
Node *lchild;
Node*rchild;
};
```

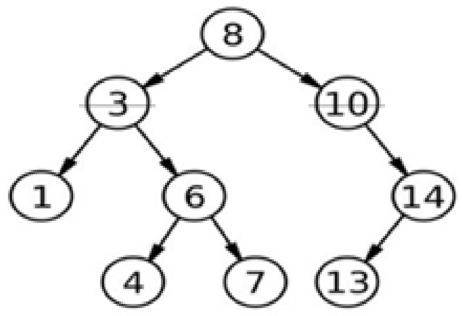
Consider a binary tree and its linked list representation is shown in the figure





5. What do you mean by the binary search tree? Write the algorithm to insert and search for anode in BST.

A binary search tree is a binary tree in which for each node in the tree, the elementsin the left subtree are less than the root and the elements in the right subtree are greater than or equal to the root.



- ***Inserting in Binary Search trees***

```
1.  Create a new BST node and assign values to it.
Node=newnode
Node->data=item
Node->left=Node->right=NULL
Return Node
2. insert(node, key)
    i) If root == NULL,
        return the new node to the calling function.
    ii) if (key>root->data)
        call the insert function with root=>right and assign the return value in root=>right.
       root->right = insert(root=>right,key)
    iii) else
        call the insert function with root->left and assign the return value in root=>left.
        root=>left = insert(root=>left,key)
3. Finally, return the original root pointer to the calling function.
```

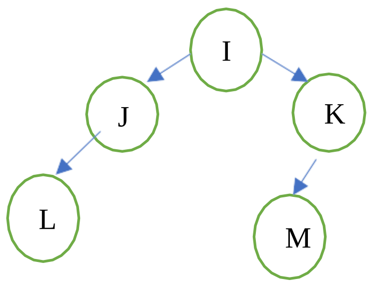
- ***Searching in Binary Search trees***

```
1.  Search (root, item)
2.  Step 1 - if (item = root → data) or (root = NULL)
3.  return root
4.  else if (item < root → data)
5.  return Search(root → left, item)
6.  else
7.  return Search(root → right, item)
8.  END if
9.  Step 2 - END
```

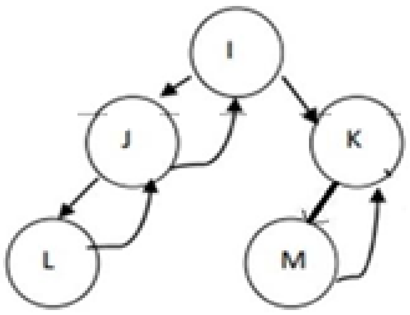
6. Explain the concept of threaded binary trees.

A threaded binary tree is a binary tree in which the nodes that do not have a right child, have a thread to their inorder successor. By doing this type of threading, we avoid the recursive method of traversing a tree, which uses stacks and also wastes a lot of memory and time.

Consider a binary tree given below.

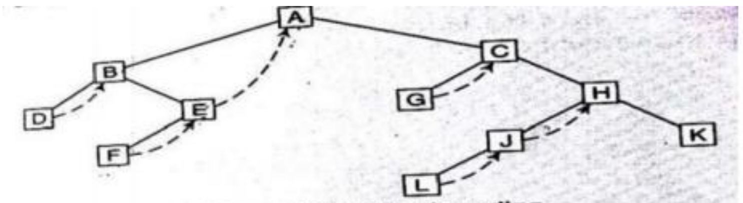


Let us make a thread binary tree out of a normal binary tree

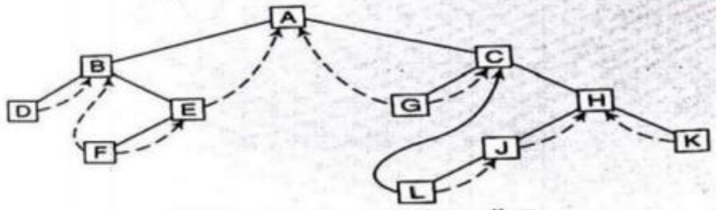


Threads; Inorder threading

- A binary tree T is maintained in memory by means of a linked list representation.
- Half of the entries in the pointer fields LEFT and RIGHT will contain null elements.
- It will replace certain null entries by special pointers which point to nodes higher in the tree.
- These special pointers are called threads and binary trees with such pointers are called threaded trees.
- The threads in threaded tree are usually indicated by dotted lines.
- There are many ways to thread a binary tree T, one may choose a one-way threading or two way threading.
- Our threading will correspond to the inorder traversal of T.
- In the one way threading of T, a thread will appear in the right filed of a node and will point to the next node in the inorder traversal of T. and in two way threading of T, a thread will also appear in the LEFT field of a node and will point to the preceding node in the inorder traversal of T.



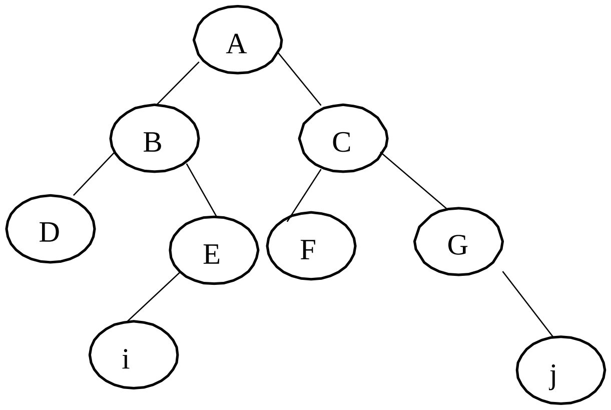
(a) One-way inorder threading



(b) Two-way inorder threading

(Questions for Application)

1. **Write the algorithm for preorder traversal of a tree.**



The preorder traversal of a non-empty binary tree is defined as follows.

1. Visit the root node
2. Traverse the left subtree in preorder
3. Traverse the right subtree in preorder

In an preorder traversal
After visiting the root node, the left subtree is taken up and it is traversed recursively, then the right subtree is traversed recursively.

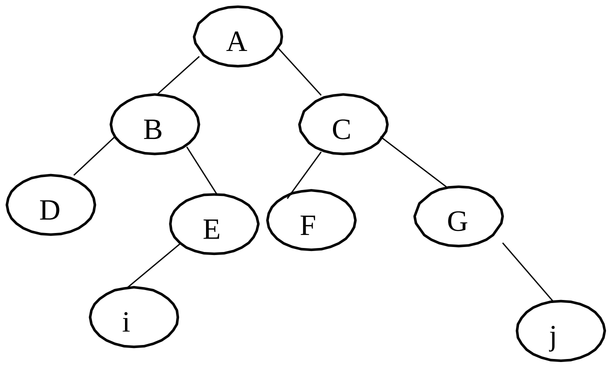
Algorithm for preorder traversal

Procedure Preorder(root)

```
1. If (root !=NULL)
2. Begin
3. Print root-> info
4. Preorder(root-> left)
5. Preorder(root-> right)
6. End
7. finished
```

The preorder traversal of the above binary tree is **ABDEICFGJ**

2. Write an algorithm for inorder traversal of a tree.



The inorder traversal of a non-empty binary tree is defined as follows.

1) Traverse the left subtree in inorder
2) Traverse the root node in inorder
3) Traverse the right subtree in inorder

In an Inorder traversal
The left subtree is traversed recursively before node. After visiting the root node, the right subtree is taken up and it is traversed recursively.

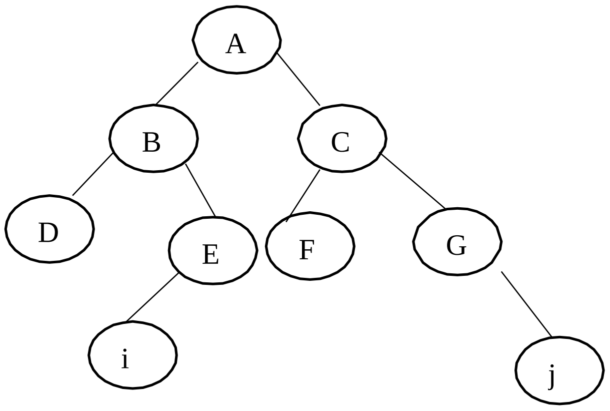
Algorithm for inorder traversal

Procedure Inorder(root)

```
1. If (root !=NULL)
2. Begin
3. Inorder(root ->left)
4. Print -> info
5. Inorder(root-> right)
6. End
7. Finished
```

The Inorder traversal of the above binary tree is **DBIEAFCGJ**

3. Write an algorithm for postorder traversal of a tree.



The Postorder traversal of a non-root node is visited before traversing its left and right subtrees.

1) Traverse the left subtree
2) Traverse the right subtree
3) Traverse the root node

In a Postorder traversal the left and right subtrees are recursively processed before visiting the root. The left subtree is taken up first and is traversed postorder. Then the right subtree is taken up and is traversed in postorder. Finally, the data at the root is displayed.

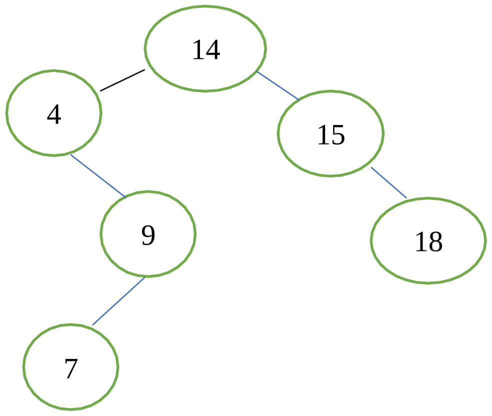
Algorithm for Postorder traversal

Procedure Postorder(root)

```
1. If (root!=NULL)
Begin
2. postorder(root ->left)
3. postorder(root ->right)
4. Print root-> info
End
5. Finished
```

The Postorder traversal of the above binary tree is **DIEBFJGCA**

**4. Construct a binary search tree for the following:
14, 15, 4, 9, 7, 18 and traverse it in Inorder, postorder and preorder.**



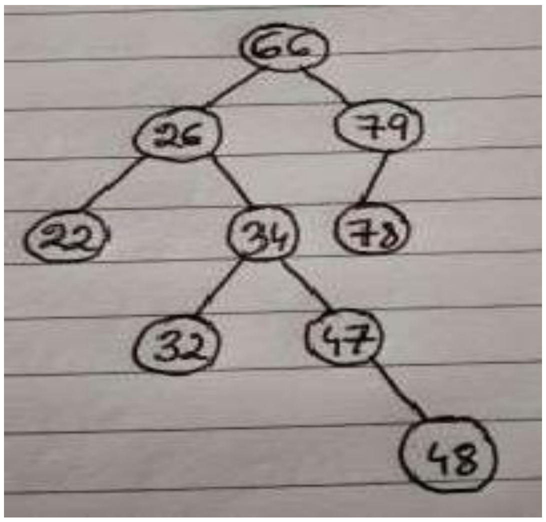
INORDER: LEFT-ROOT-RIGHT
4 7 9 14 15 18

PREORDER: ROOT-LEFT-RIGHT
14 4 9 7 15 18

POSTORDER: LEFT-RIGHT-ROOT
7 9 4 18 15 14

5. Construct a binary search tree for the following data.
66, 26,22,34,47,79,48,32,78

A binary search tree is a binary tree in which for each node in the tree, the elementsin the left subtree are less than the root and the elements in the right subtree are greater than or equal to the root.



MODULE 5

Multiple Choice Questions

(Questions for Application)

1. The worst case occurs in linear search algorithm when_________________

A. item is somewhere in the middle of the array
B. item is not in the array at all
C. item is the last element in the array
D. item is the last element in the array or item is not there at all

2. Sorting algorithm can be characterized as_________________

A. Simple algorithm which requires the order of n2 comparisons to sort n items.
B. sophisticated algorithms that require the o(n log2n) comparisons to sort items.
C. both of the above
D. none of the above

3. State true or false for internal sorting algorithms.
   i) Internal sorting are applied when the entire collection if data to be sorted is small enoughthat the sorting
   can take place within main memory.
   ii) The time required to read or write is considered to be significant in evaluating theperformance of internal sorting.

A.i-true,ii-true
B.**i-true,ii-false**
C.i-false,ii-true
D.i-false,ii-false

4. Is putting an element in the appropriate place in a sorted list yields a larger sorted
order list.A.**insertion**
B.extraction
C.selection
D.distribution

5. ________is rearranging pairs of elements which are out of order, until no such pairs remain.

A. insertion
B. exchange
C. selection
D. distribution

6.Which of the following sorting algorithm is of divide and conquer type?
A. Bubble sort
B. Insertion sort
C. Merge sort
D. Selection sort

7. ______________________partitions the given set of elements each time to find an element.
A.**Binary search**
B.Linear search
C.Quick search
D.Merge search

8. The elements in the array must be in___________________order to perform binary search.A.Unsorted

B.**Sorted**

C.Linear

D.Non-linear

9. Which sorting technique compares the elements that are distant apart?

A.Selection sort

B.Shell sort

C.Insertion sort

D.Quick sort

(Questions for Skill)

10. If the number of records to be sorted is small, then________________sorting can be efficient.

A.merge

B.heap

C.selection

D.bubble

11. Which of the following is not a limitation of binary search algorithm?

A. must use a sorted array

B. requirement of sorted array is expensive when a lot of insertion and deletions are needed

C. there must be a mechanism to access middle element directly

D. binary search algorithm is not efficient when the data elements more than 1500

12. The average case occurs in linear search algorithm_________________

A. **when item is somewhere in the middle of the array**

B. when item is not the array at all

C. when item is the last element in the array

D. item is the last element in the array or item is not there at all

13. Binary search algorithm cannot be applied to____

A. **sorted linked list**

B. sorted binary trees

C. sorted linear array

D. pointer array

14. Sorting algorithm is frequently used when n is small where n is total number of elements.

A.heap

B.insertion

C.bubble

D.quick

15. Which of the following sorting algorithm is of priority queue sorting type?

A.Bubble sort

B.Insertion sort

C.Merge sort

D.Selection sort

16. Which of the following is not the required condition for binary search algorithm?

A. the list must be sorted
B. there should be the direct access to the middle element in any sub list
C. there must be mechanism to delete and/or insert elements in list
D. number values should only be present

17. Partition and exchange sort is__________________

A. **Quick sort**
B. Tree sort
C. Heap sort
D. Bubble sort

18. ___________is technique that performs search process in a sequential manner.

A.Linear search
B.Binary search
C.Hashing
D.Indexing

19. Which of the following is true?

A.A graph may contain no edges and many vertices
B.A graph may contain many edges and no vertices
C.A graph may contain no edges and no vertices
D.None of the mentioned

20.A connected graph T without any cycles is called a___________________

A.A tree graph
B.Free tree
C.A treed
D.All of the above

21. In a graph if E=(u,v)means_____________________

A. u is adjacent to v but v is not adjacent to u
B. e begins at u and ends at v
C. u is processor and v is successor
D. both b and c

22.A path in a digraph in which all the edges are distinct is called ________________________

A.Simple path
B.Elementary path
C.Cycle
D.Loop

23. Any two nodes that are connected by an edge in a graph are called______________________ nodes

A.Directed
B.Adjacent
C.Common
D.Isolated

24. In a graph if e=[u,v], Then u and v are called_____________________

A.End points of e
B.Adjacent nodes
C.Neighbours
D.All of the above

25.A graph in which some edges are directed and some edges are undirected is called______graph.
A.Digraph
B.Mixed
C.Isolated
D.Cycle

Questions carrying 4 marks

(Questions for Application)

1. Write and explain the algorithm for the insertion sort technique.

The algorithm of insertion sort functions as follows. Initially to start the whole array is in a completely unordered state. The first element is considered to be in the ordered list. The second element is considered to be in the unordered list. The second element is then inserted either in the first or in the second position as appropriate. Now there are 2 elements in the ordered part and remaining elements are considered to be unordered. Inserting the third element, then the fourth and so on slowly extends the ordered part.

Pass 1: A[1] by itself is sorted because it is the first element
Pass 2: A[2] is inserted either before of after A[1] so that now A[1] , A[2] are sorted.
Pass 3: A[3] is inserted into its proper place in A[1], A[2] that is, before A[1], between A[1]and A[2] or after A[2] so that A[1], A[2], a[3] are sorted.
Pass 4: A[4] is inserted into its proper place in A[1], A[2], A[3] so that: A[1], A[2], A[3] and A[4] is sorted.

Algorithm I

Procedure InsertionSort(A, N)

```
This algorithm sorts an A with N elements
Step 1: repeat step 2 for i=1 to N-1
Step 2: repeat step 3 for j=I to 0
Step 3: if A[J] < A[J-1]
          begin
             a. set TEMP:=A[J]
             b. set A[J]:=A[J-1]
             c. set A[J-1]:=TEMP
[end of if structure]
[end of inner for loop]
[end of outer for loop]
```

OR

Algorithm II

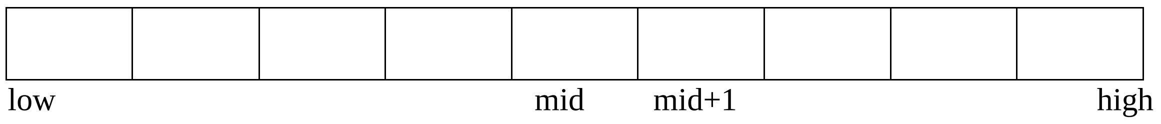
Given an array A of N an element, this procedure sorts the element in theascending order. The variables I and J are used to index the array elements.

Procedure Insertion _Sort (A, N)

```
Step 1:For I =1 to N do
Step 2: TEMP=A[I]
Step 3: J=1-1
Step 4: While (J>=0) and (A[J]>TEMP)
Step 5: A[J+1]=A[J]
Step 6: J=J-1
[End of while loop]
Step 7: A[J+1]=TEMP
[end of step 1 for loop]
Step 8: Exit
```

2. Write the algorithm for merge sort method.

In this technique, a single array is divided into two sub lists. The two sub lists are sorted. i.e. elements from low to mid are sorted and elements from mid+1 to high are sorted. But, the elements from low to high are unsorted. The efficiency of merge sort is $O(n \log_2 n)$.



Hence we merge the two sorted sub lists into a single sorted array. If low and high are lower and upper limits in an array then, the general procedure to implementmerge sort is as follows:

If(low≤high)

1. Divide the array into equal parts
2. Sort the left part of the array recursively
3. Sort the right part of the array recursively
4. Merge the left and right parts
5. End if

Algorithm:

Algorithm MergeArray(A, LOW,MID,HIGH). This algorithm merges two sortedarrays where the first array is from LOW to MID and the second array is from MID+1 to HIGH.

```
1.  I←LOW, J←MID+1, K←LOW
2.  While (I≤MID and J≤HIGH)
       repeat
3.  If (A[I]<A[J])
    Then C[K] ←A[I], I←I+1,
    K←K+1 Else C[K]=A[J],
    J←J+1,K←K+1
4.  While(I≤MID) repeat steps 5,6
5.  C[K] ←A[I]
6.  K←K+1, I←I+1
7.  While(J≤HIGH) repeat steps 8,9
8.  C[K] ←A[J]
9.  K←K+1, J←J+1
10. For I=LOW to HIGH repeat step 11
11. A[I] ←C[I]
12. Return.
```

Algorithm for Merge Sort:

Algorithm MergeSort(A, LOW, HIGH). The purpose of this algorithm is to sort the elements of array A between the lower and upper bounds LOW and HIGH respectively.

```
    1.  If(LOW≠HIGH) then perform steps 2-5
```

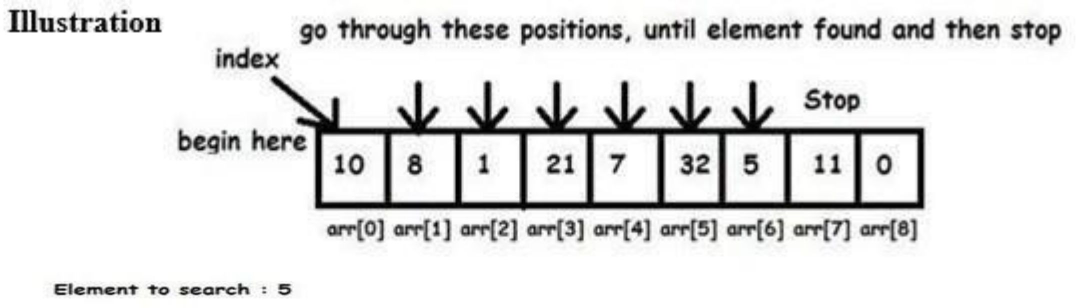
```
2.  MID←(LOW+HIGH)/2
3.  MergeSort(A,LOW,MID)
4.  MergeSort(A,MID+1,HIGH)
5.  MergeArray(A,LOW,MID,HIGH)
6.  Return
```

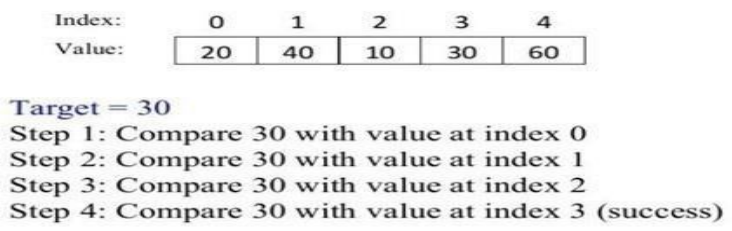
3.Write the linear search algorithm with an example.

Algorithm Linear_Search(A,element,N)

A is an array of N elements and element is the element being searched in the array.

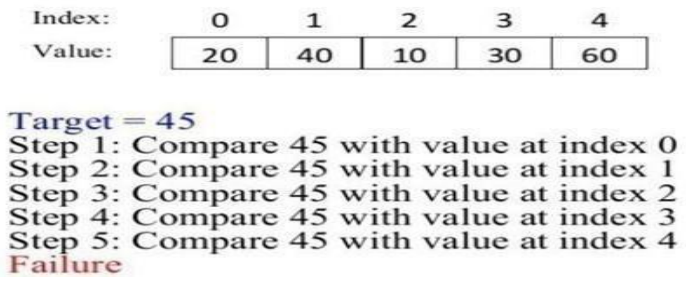
```
Step 1: Set Loc:=-1
Step 2: Repeat step3 For i=0 to n-1
Step 3: If (Element=A[i]) then
            begin
              i.  Set loc:= i
              ii. Goto step 4
              [End if]
              [End for loop]
Step 4: If (loc>=0) then
              i.   Write("Element found at location,loc+1)
              ii.  Else
              iii. Write("Element not found")
Step 5:Exit
```





Index:	0	1	2	3	4
Value:	20	40	10	30	60

Target = 30
Step 1: Compare 30 with value at index 0
Step 2: Compare 30 with value at index 1
Step 3: Compare 30 with value at index 2
Step 4: Compare 30 with value at index 3 (success)



Index:	0	1	2	3	4
Value:	20	40	10	30	60

Target = 45
Step 1: Compare 45 with value at index 0
Step 2: Compare 45 with value at index 1
Step 3: Compare 45 with value at index 2
Step 4: Compare 45 with value at index 3
Step 5: Compare 45 with value at index 4
Failure

4. Write the binary search algorithm with an example.

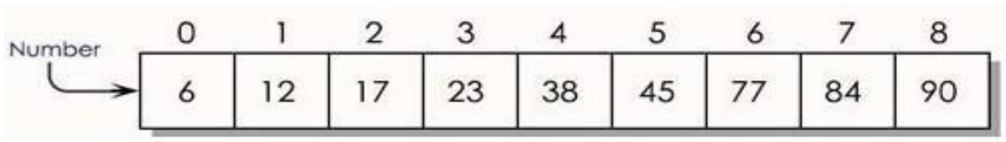
Algorithm

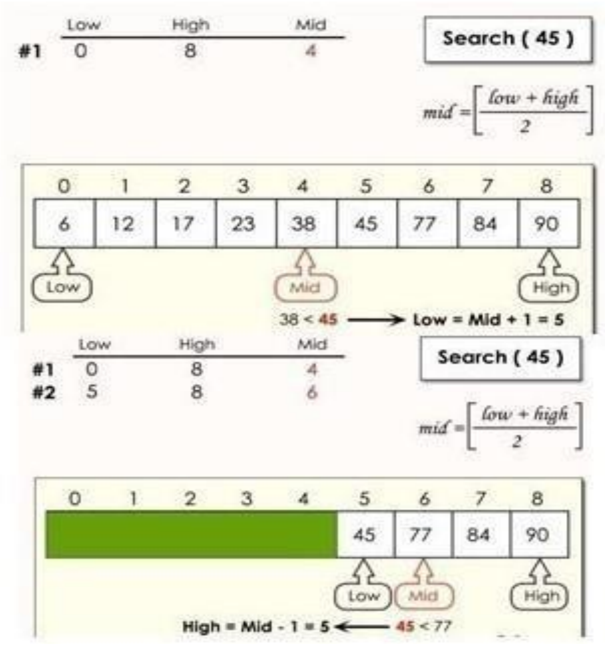
Binary_search(A, element, N)

A is a list of N elements and the element is the element being searched in the array. LOW and HIGH identify the positions of the 1st and last elements in a range and MID identifies the position the middle element.

```
Step 1: Set LOW:= 0
 Step 2: Set HIGH:= N-1
Step 3: Set LOC:= -1
Step 4: Repeat steps 5 and 6, 7
            While LOW<=HIGH
           Begin
Step 5: Set MID:=LOW+HIGH)/2
Step 6: IF element=A[MID]
            Begin
                 a. Set LOC:= MID
               goto step 8
             [END IF]
Step 7: IF element<A[MID]
                 i.      Set HIGH:=MID-1
                 ii.     Else
                 iii.    Set LOW:=MID+1[END IF]
             [END of While loop]
Step 8:Write("Element found in location", LOC)
             ELSE
            Write("Element not found")[END IF]
Step 9: EXIT
```

IILUSTRATION:





5. Write an algorithm for breadth first search (BFS).

Breadth First Search (BFS)

This is a very different approach for traversing the graph nodes. The aim of BFS algorithm is to traverse the graph as close as possible to the root node. Queue is used in the implementation of the breadth first search.

<u>Algorithmic Steps</u>

Step 1: Push the root node in the Queue.
Step 2: Loop until the queue is empty.
Step 3: Remove the node from the Queue.
Step 4: If the removed node has unvisited child nodes, mark them as visited andinsertthe unvisited children in the queue.

6. Write an algorithm for depth first search (DFS).

Depth First Search (DFS)

The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node. Stack is used in the implementation of the depth first search.

<u>Algorithmic Steps</u>

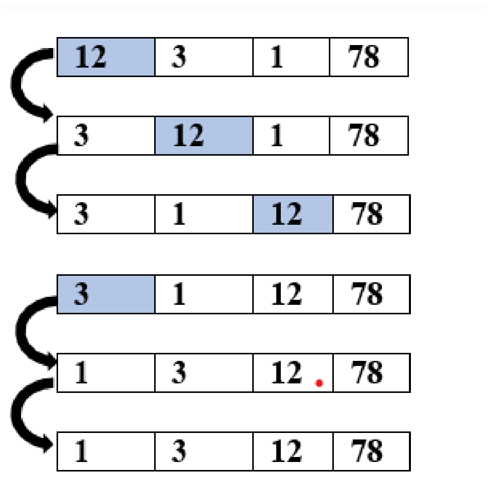
Step 1: Push the root node in the Stack.
Step 2: Loop until stack is empty.
Step 3: Peek the node of the stack.(Peek used to return the value of top)
Step 4: If the node has unvisited child nodes, get the unvisited child node, mark it astraversed andpush it on stack.

Step 5: If the node does not have any unvisited child nodes, pop the node from the stack.

(Questions for Skill)

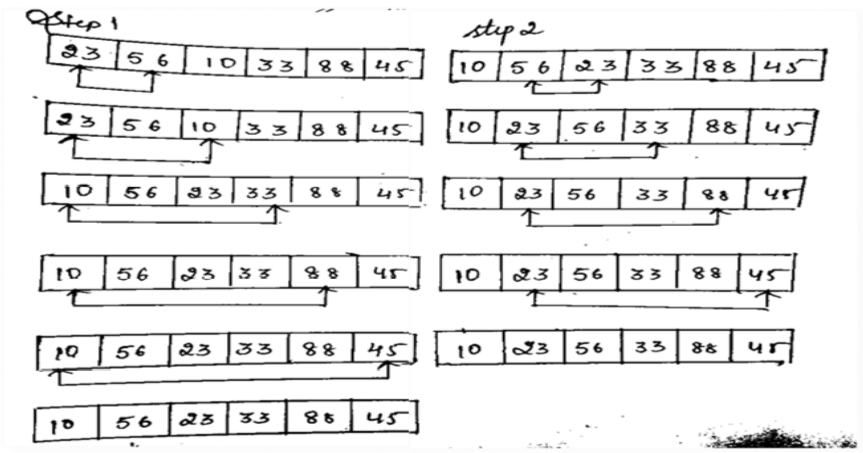
1.Sort the following numbers using insertion sort method.

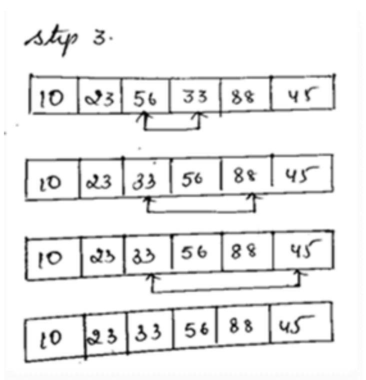
12 3 1 78

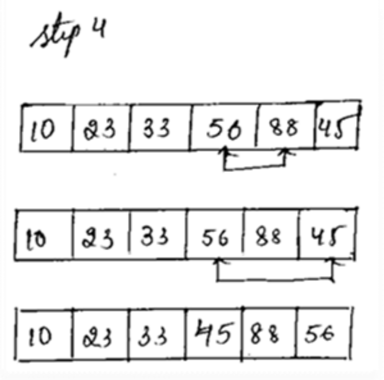


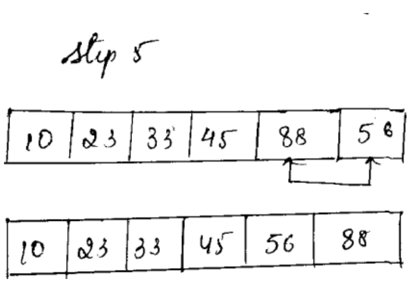
2. Sort the following numbers using selection sort technique.

23, 56, 10, 33, 88, 45



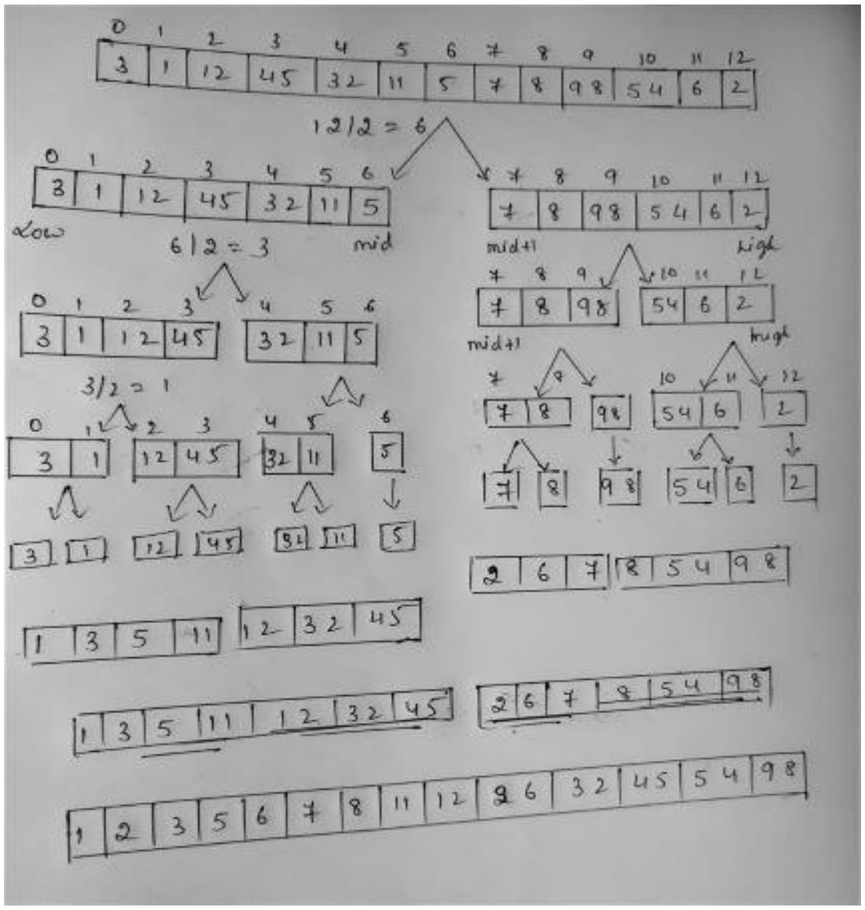






3. **Sort the following numbers using merge sort.**

3 1 12 45 32 11 5 7 8 98 54 6 2



4. **Write and explain the algorithm for bubble sorting procedure.**

Bubble Sort:

Step 1: compare A[1] and A[2] and arrange them in the desired order so that A[1] < A[2]. Then compare A[2] with A[3] and arrange them so that A[2]< A[3]. Continue until we compare A[N-1] with A[N] and arrange them so that A[N-1] < A[N]. During step 1 the largest element is bubble up to the A[N] position. And step 1 involves n-1 comparisons.

Step 2: repeat step 1 with one less comparison; that is now we stop after we compare and possibly arrange A[N-2] and A[N-1] at the end of step 2, the second largest element in the array will occupy the A[N-1] position.

Step N-1: compare A[1] with A[2] and arrange them so that A[1] < A[2]. After step N-1 steps, the list will be sorted in increasing order.

Algorithm

Given an array A of N elements, this procedure sorts the elements in the ascending order using the method described above. The variable I and J are used to index the array elements.

```
Bubble_Sort (A, N):

Step 1: for I= 1 to N-1 do
    Step 2: for J = 0 to N-I-1 do
        Step 3: [Compare adjacent elements]
            If A[J] > A[J+1] then
                Step 4: Temp = A[J]
                Step 5: A[J] = A[j+1]
                Step 6: A[J+1] = Temp
            [End If]
    [End of Step 2 for loop]
[End of Step 1 for loop]
Step 7: Exit
```

5. Write the algorithm for selection sort method.

Pass 1: find the location POSITION of the smallest element in the list of N elements and then interchange A[POSITION] and A[1], then A[1] is sorted

Pass 2: find the location POSITION of the smallest element in the sublist of N-1 elementsand then interchange A[POSITION] and A[2] then, A[1] and A[2] are sorted since A[1]
<=A[2]

Pass 3: find the location POSITON of the smallest element in the sublist of N-2 elements and then interchange A[POSITION] and A[3] then, A[1], A[2], A[3] is sorted since A[2]<=A[1]. And so on. Thus A is sorted after N-1 passes.

Algorithm

A is an array of N elements. This algorithm finds the smallest element SMALL and its location POSITION among the elements in the array A.

```
Procedure SelectionSort(A, N)

Step 1: repeat steps 2, 3 and 4, 6, 7 for i=1 to N-1
    begin
    Step 2: set SMALL:=A[I]
    Step 3: set POSITION:=I
    Step 4: repeat step 5 for J=I+1 to N
        begin
        Step 5: if A[J] < SMALL
            begin
            a.set SMALL:=A[J]
            b.set POSITION:=J
        [End of if structure] [end of inner for loop]
```

```
Step 6: Set A[POSITION]:=A[I]
Step 7: Set A[I]:=SMALL
[end of outer for
loop]Step 8: Exit
```

6. **Write the algorithm for the Quick sort method.**

Algorithm:

```
1.  QUICKSORT (array A, start, end)
2.  {
3.    if (start < end)
4.    {
5.         p = partition(A, start, end)
6.         QUICKSORT (A, start, p - 1)
7.         QUICKSORT (A, p + 1, end)
8.     }
9.  }
```

PARTITION ALGORITHM:

```
1.  PARTITION (array A, start, end)
2.  {
3.    pivot = A[end]
4.    i = start-1
5.    for j = start to end -1
6.  {
7.    do if (A[j] < pivot)
8.   {
9.    then i = i + 1
10.   swap A[i] with A[j]
11.   }
12. }
13.   swap A[i+1] with A[end]
14.  return i+1
15. }
```

OR

Partition(A, LOW,HIGH) This function partitions the array A with LOW and HIGH as lower bound and upper bound respectively into two sub lists.

```
1. KEY←A[LOW], I←LOW+1, J←HIGH
2. Repeat steps 3-7
3. Repeat step 4 while(I<HIGH and KEY≥A[I])
4. I←I+1
5. Repeat step 6 while (KEY<A[J])
6. J←J-1
7. If(I<J)
```

```
    Then A[I]↔A[J]
    Else A[LOW]↔A[J] , Return J
8.  Return
```

Algorithm QuickSort(A, LOW, HIGH). This is a recursive algorithm to sort the elements in array A with LOW and HIGH as lower and upper bound respectively.

```
1.  If (LOW<HIGH) repeat steps 2-4
2.  J←Partition(A,LOW,HIGH)
3.  QuickSort(A,LOW,J-1)
4.  QuickSort(A,J+1,HIGH)
5.  Return
```