Name : Mazhar Saleem
Id : 14455 cs4th

Q1:- How would you define a linked list?

Ans:- **Definition:-**

A linked list is a sequence of data structures, which are connected together via links. Linked lists is a sequence of links which contains items, Each link contains a connection to another link.

OR

It is a data structure consisting of a collection of nodes which together represent a sequence. each node contains data and a reference (in other words a link).

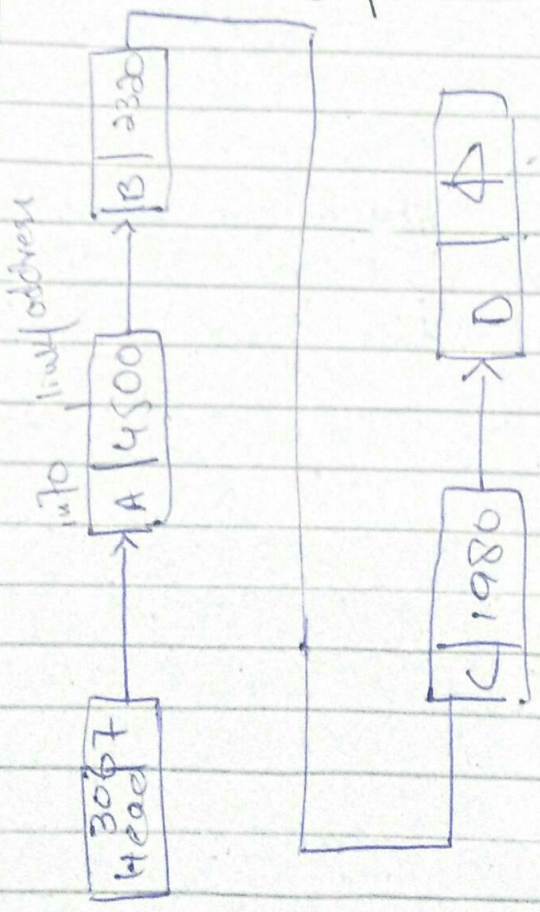
Explanation:-

Linked lists are among the simplest and most common data structures, They can be used to implement several other common abstract data types, including lists, stacks, queues though it is not uncommon to implement those data structures directly without using a linked list as the basis." A linked list is a list whose elements may not occupy continuous

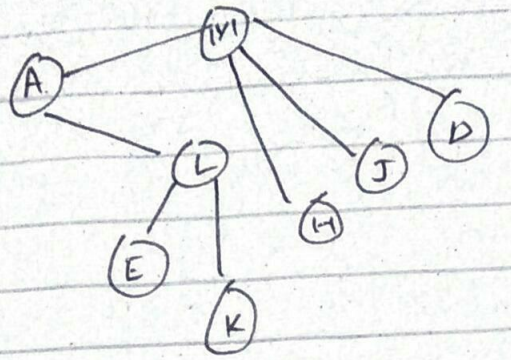
memory locations and whose elements are connected by means of links between them.

* Each element of a linked list is called a node.
* Each node has at least 2 fields
  - i:- info field :- keeps data
  - ii:- link field :- keep address of next node.
* A pointer "Head" is used to keep the address of 1st node.

(Q) b) Design a diagramatic one way linked list for the given data:

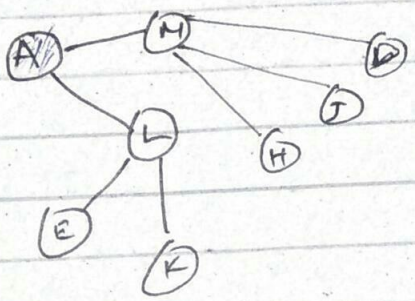


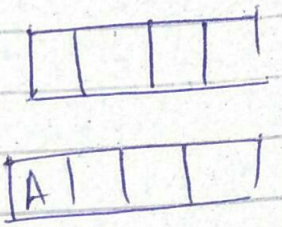
Q2 Apply depth-first technique:



Step 1:-

* An empty stack is taken, starting from root node "A".
* push to stack and mark as visited.

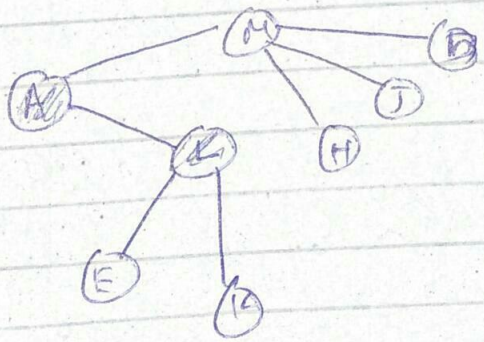


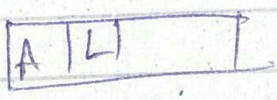


Output sequence = A

Step 2:-

* A is adjacent to M and L
* Follow alphabetically. select "L".
* Highlight L and push into stack.

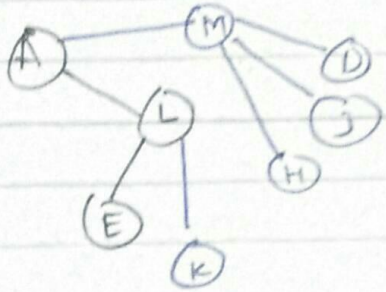




O.S = A.L

Step 3:-

* L is adjacent to E and K
* Follow alphabetically sorted "E".
* Highlight "E" as visited
* push E to the stack.



A	L	E	

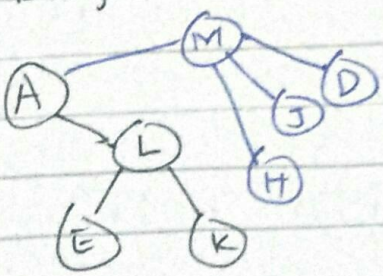
∴ Output sequence:

A , L , E

Step 4:-

* As "E" is a leaf node so "E" is popped from top of the stack, and get back to "L"
* K is adjacent to "L"
* So K is on top of the stack.

Diagram:

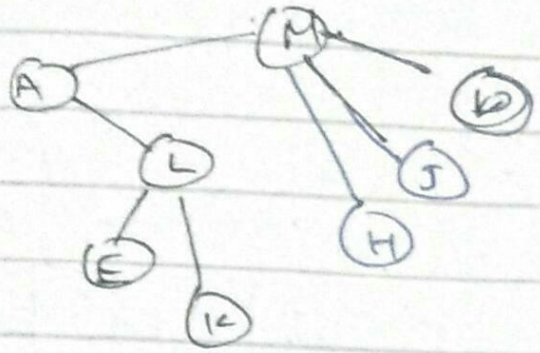


A	L	~~E~~	K

∴ O/S:- A, L, E, K.

As K is a leaf node so K is popped from top of the stack and M is pushed to the top of the stack

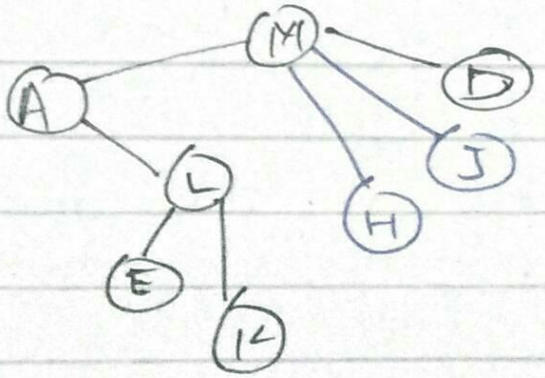
A	L	~~K~~ M	



O·S:- A, L, E, K, M

Step 6:-

* As M is adjacent to D and J
* Follow alphabetically select "D".
* Higlight "D".
* push D to the stack.

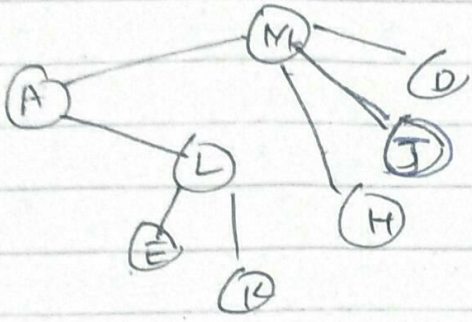


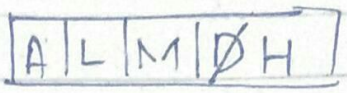
A	L	M	D

O·S:- A, L, E, K, M, D

Step 7:-

* As D is a leaf node so it is popped from Stack. and pushed back to M.
* M is adjacent to H and J
* Follow alphabetically put "H".
* Mark and push H as visited into the Stack.

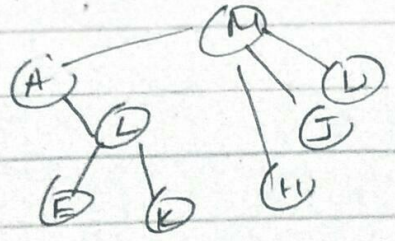


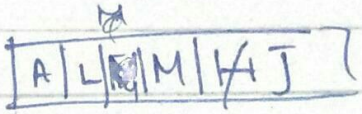


O·S:- A L E K M D H

Step 8:-

* As H is a leaf node so it is popped from stack. and back to M, and J as adjacent to M.
* Mark J as visited
* push J to the top of the stack





A, L, E, K, M, D, H, J

(3) How would you be defining a Queue? Give some real life example.

Queue:-

A sequential list in which elements are inserted from one end and are deleted/retrieved from other end is called Queue.

Explanation:-

Queue is an abstract data structure, A Queue is open at both its ends. one end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First in First out methodology, i.e the data item stored first will be accessed first.

* The End from where an element can be inserted is called Rear of the Queue.
* The End from where an element can be deleted called Front of the Queue.

Memory representation of Queue

* A Linear array q[] is used to represent a Queue.
* Two variables (behaving like

pointers) F and R are used to denote Front and Rear of q[ ].

Real life examples:-

(i) Automobiles waiting to pass through a signal make up a queue.

(ii) People waiting to submit bills at a bank window.

(iii) Another real-world example of Queue can be a single-lane one way road, where the vehicle enters first, exit first. More real-world examples can be seen as queues at the ticket windows and bus stops.

(iv) ~~Stack of plates in a cupboard.~~

(v) ~~Their Stack of~~

Q1b Further explanation (execution)

(1) p ← get node(3067)
(2) Head ← p
(3) info(p) ← date

3067 ← A
4500 ← B
2320 ← C
1980 ← D

Link (p) ← (p
(3067)(4500)(2320)(1980)

(5) q ← p (3067)(4500)(2320)(1980)

(6) Yes Yes Yes Yes No

(7) p ← get node (4500)(2320)(1980)

(8) Link (q) ← p (4500)(2320)(1980)

(9) go to 3

(10) Exit